kMesh: Un Algoritmo Paralelo para Construir Mallas Adaptativas a partir de Imágenes

Ronald Ubel Adolfo Gonzales Vega

Orientador: Dr. Alex Jesús Cuadros Vargas

Jurado:
Dr. James Gee – University of Pennsylvania – USA
Dr. Luis Gustavo Nonato – Universidade de Sao Paulo – Brasil
Dr. Javier Montoya – Swiss Federal Institute of Technology – Switzerland
Dr. José Eduardo Ochoa Luna – Universidad Católica San Pablo – Perú

Tesis presentada al Departamento de Ciencia de la Computación
como parte de los requisitos para obtener el grado de
Maestro en Ciencia de la Computación.

Universidad Católica San Pablo – UCSP
Diciembre de 2018 – Arequipa – Perú

A mi familia, por todo lo que me ha dado.

Abreviaturas

CPU *Central Processing Unit*

GPU *Graphic Processing Unit*

FMM *Fast Marching Method*

FIM *Fast Iterative Method*

FSM *Fast Sweeping Method*

GNG *Growing Neural Gas*

GVF *Gradient Vector Flow*

CVT *Centroidal Voronoi Tessellation*

AOF *Average Outward Flux*

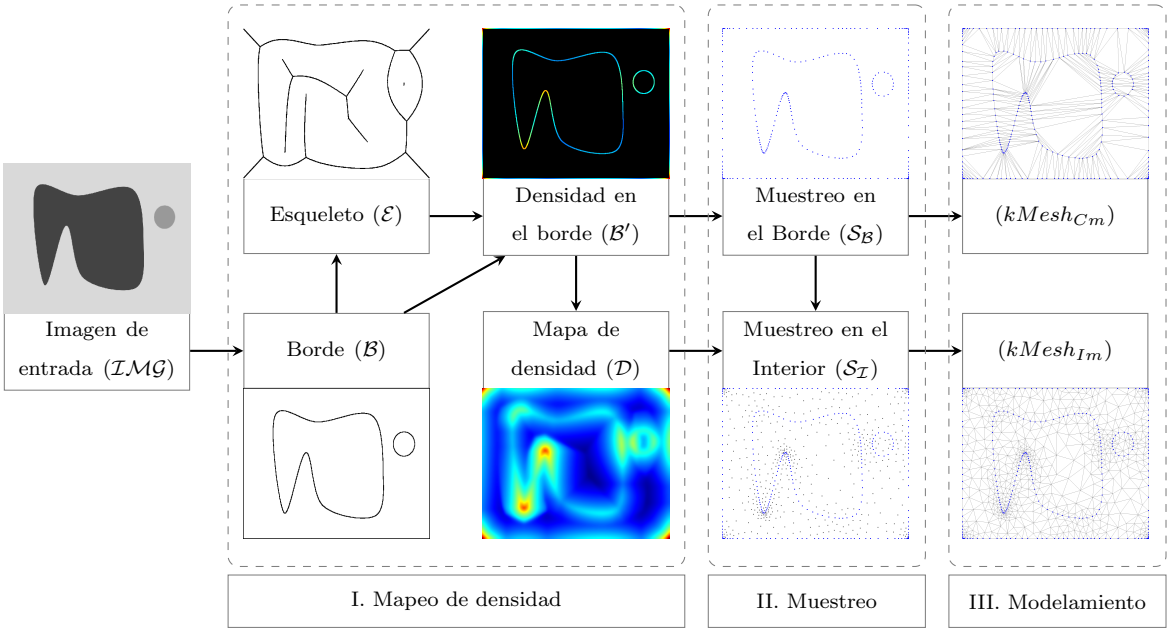
Agradecimientos

En primer lugar, deseo agradecer a Dios por haberme guiado a lo largo de estos dos años de estudio.

Agradezco a mis padres y a mi familia por el apoyo brindado para forjarme como persona y profesional.

Deseo agradecer al Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC) y al Fondo Nacional de Desarrollo Científico, Tecnológico e Innovación Tecnológica (FONDECYT-CIENCIACTIVA), que mediante Convenio de Gestión UCSP-FONDECYT Nº011-2013, han permitido la subvención y financiamiento de mis estudios de Maestría en Ciencia de la Computación en la Universidad Católica San Pablo (UCSP).

Abstract



With the development of graphic computing methods and technologies that allow capturing volumetric images an important development was made to generate geometric models, among them $Imesh$method, which is an algorithm that constructs simplicial meshes from non-preprocessed images in 2 and 3 dimensions. $Imesh$ is divided into 3 stages: Construction ($Imesh_{Cm}$), of a Delaunay mesh from an input image; Partitioning ($Imesh_{Pm}$), of the mesh in a defined number of sub-meshes, using its geometric and topological information; and Improvement ($Imesh_{Mm}$), of the elements that make up the sub-meshes generated by introducing Delaunay mesh quality criteria. This work studies and reformulates the stages of Construction ($Imesh_{Cm}$) and Improvement ($Imesh_{Mm}$) of the $Imesh$ method, and uses this analysis to propose a new mesh construction method, named $kMesh$. This new idea uses a combination of distance map, skeletonization and adaptive Poisson disk sampling. This way, our work proposes a parallel algorithm, to produce adaptive meshes from images, in 2 and 3 dimensions, considering quality criteria in the generated elements.

Keywords: Parallel Programming, Image Mesh Construction, Poisson Sampling.

Resumen

Con el desarrollo de métodos de computación gráfica y tecnologías que permiten captar imágenes volumétricas, se abrió paso a un desarrollo importante de métodos para generar modelos geométricos, entre ellos, se encuentra el método $Imesh$, el cual es un algoritmo que construye mallas simpliciales a partir de imágenes no preprocesadas, en 2 y 3 dimensiones. $Imesh$ está dividido en 3 etapas: Construcción ($Imesh_{Cm}$), de una malla de Delaunay a partir de una imagen de entrada; Particionamiento ($Imesh_{Pm}$), de la malla en un número definido de submallas, usando su información geométrica y topológica; y Mejoramiento ($Imesh_{Mm}$), de los elementos que componen las submallas generadas introduciendo criterios de calidad de mallas Delaunay. Este trabajo estudia y reformula las etapas de Construcción ($Imesh_{Cm}$) y Mejoramiento ($Imesh_{Mm}$) del método $Imesh$, y utiliza este análisis para proponer un nuevo método de construcción de mallas, denominado $kMesh$. Esta nueva idea utiliza una combinación de mapas de distancia, esqueletización y distribución adaptativa de puntos con discos de Poisson. De esta manera, nuestro trabajo propone un algoritmo paralelo, para producir mallas adaptativas a partir de imágenes, en 2 y 3 dimensiones, considerando criterios de calidad en los elementos generados.

Palabras clave: Programación Paralela, Construcción de malla, Muestreo de Poisson.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

La computación gráfica es una área que estudia métodos y técnicas para crear, manipular y analizar escenarios reales o virtuales digitalmente. El objetivo de esta área no sólo se enfoca en visualizar o generar imágenes en computadora, sino también representar y modelar problemas reales sobre distintos escenarios. Por esta razón, esta área, se ve envuelta en diversos campos tales como el tratamiento de imágenes, robótica, aeronáutica, geografía, medicina y entretenimiento (Bu-Qing y Ding-Yuan, 2014; Shirley et al., 2015).

Por otro lado, hay mucho esfuerzo en la comunidad científica para desarrollar tecnologías no invasivas como tomografías computarizadas (Jermyn et al., 2013), resonancias magnéticas (Menini et al., 2012), y microscopías electrónicas (Yuan et al., 2015), las cuales permiten captar imágenes volumétricas de objetos reales, que muchas veces son de origen biológico (Tian et al., 2014). Con esta información volumétrica y el desarrollo de métodos de computación gráfica, se abrió paso a un desarrollo importante de métodos para generar modelos geométricos a partir de imágenes, aplicados en simulaciones quirúrgicas (Mostaghimi et al., 2014), simulaciones numéricas de fenómenos físicos o biológicos (Lu et al., 2014), y en el análisis virtual no invasivo de estructuras internas (Silva Vieira et al., 2015).

Por lo anterior, surgió una gran variedad de métodos enfocados en la generación de mallas a partir de imágenes en 2 y 3 dimensiones. La mayoría de estos métodos, necesitan de un paso previo de preprocesamiento (Sonka et al., 2014), para eliminar el ruido proveniente de la imagen, o para segmentar los objetos que se encuentran en la imagen. Por otro lado, dentro de toda esta variedad de métodos, surgió un algoritmo denominado $Imesh$(Cuadros-Vargas et al., 2009), que construye mallas simpliciales a partir de imágenes no preprocesadas, en 2 y 3 dimensiones. Este método divide su procedimiento en 3 etapas principales (ver Figura 1.1): **Construcción** de la malla ($Imesh_{Cm}$), produce una malla de Delaunay a partir de una imagen de entrada añadiendo patrones de color a cada elemento generado; **Particionamiento** de la malla ($Imesh_{Pm}$), usa información geométrica y topológica de la malla para dividirla en un número definido de submallas; y **Mejoramiento** de calidad de la malla ($Imesh_{Mm}$), mejora los elementos de las submallas generadas en el interior de la imagen, agregando criterios de calidad de mallas Delaunay (Shewchuk, 2002; Belytschko, 2008).

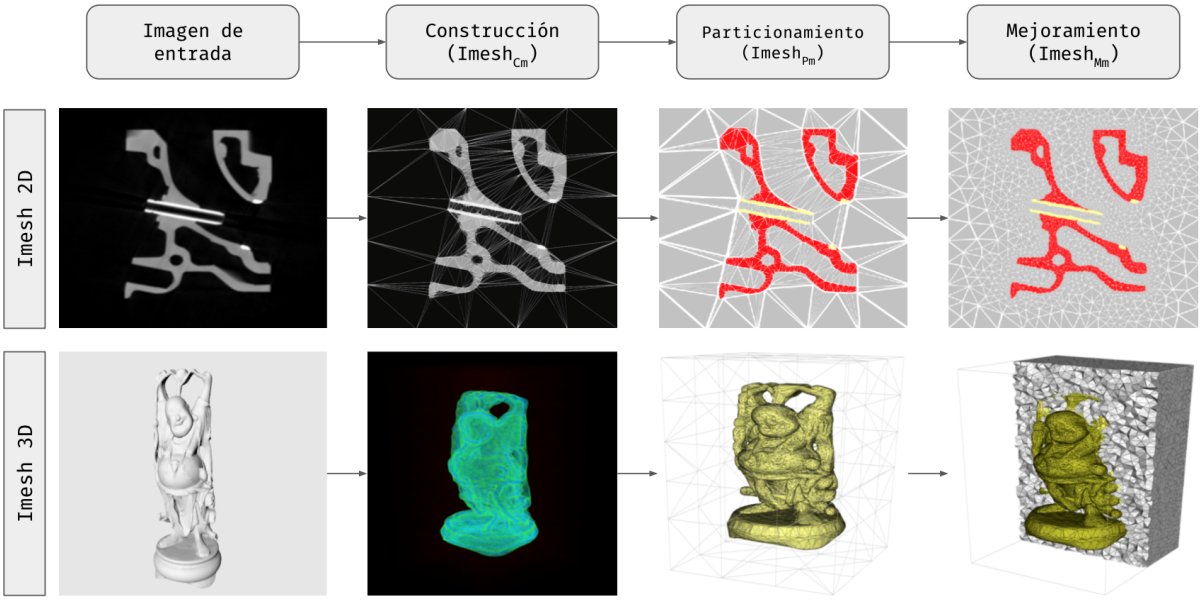


Figura 1.1: (Cuadros-Vargas et al., 2009). Etapas del método *Imesh* a partir de una imagen en 2 y 3 dimensiones: Construcción ($Imesh_{Cm}$), genera una triangulación de Delaunay; Particionamiento ($Imesh_{Pm}$), subdivide en submallas considerando información geométrica y topológica; y Mejoramiento ($Imesh_{Pm}$), mejora los criterios de calidad de las submallas generadas en el interior de la imagen.

Este trabajo se enfoca en analizar la etapa $Imesh_{Cm}$ para proponer un nuevo método de construcción de mallas a partir de imágenes, denominado $kMesh$.

Durante el análisis realizado sobre el método *Imesh*, surgieron algunas observaciones, descritas a continuación. Primero, la etapa $Imesh_{Cm}$, se basa en un procedimiento secuencial para construir la malla, en 2 y 3 dimensiones, por lo que es bastante dependiente, en cuanto a tiempo de ejecución, de las dimensiones de la imagen de entrada. Segundo, $Imesh_{Cm}$, es un proceso que inserta puntos minimizando una función de aproximación de la malla hacia los bordes de la imagen. Sin embargo, esta aproximación no considera la curvatura de estos bordes, por esta razón, puede haber regiones con mucha curvatura y pocos puntos, o regiones con poca curvatura con más puntos de los necesarios. Por último, la etapa $Imesh_{Mm}$, en 2 dimensiones, garantiza la calidad de una malla Delaunay. En 3 dimensiones, este método no garantiza una malla con criterios de calidad de Delaunay.

Considerando los aspectos mencionados anteriormente sobre el método *Imesh*, nuestro algoritmo propuesto, $kMesh$, tiene las siguientes características:

1. Es un algoritmo paralelo de construcción de mallas, extensible de 2 a 3 dimensiones.
2. Distribuye puntos sobre las características de la imagen considerando criterios de curvatura y proximidad entre objetos, con el objetivo de producir una malla adaptativa.
3. Produce una nube de puntos ajustados a las características de la imagen de tal

manera que al generar una triangulación de Delaunay con esta nube, esta malla contenga criterios de calidad.

El resto de esta investigación está organizada de la siguiente manera. En el Capítulo 2 se presentan los métodos más relacionados a este trabajo, considerando que sean extensibles de 2 a 3 dimensiones y puedan ser implementados de forma paralela. Para obtener información acerca de la forma de un objeto, se estudian los mapas de distancia y los métodos de esqueletización. Para distribuir puntos de forma adaptativa sobre una imagen, se investigan los algoritmos de muestreo de discos de Poisson. En el Capítulo 3 se definen los algoritmos utilizados en nuestra propuesta. En el Capítulo 4 se describe la propuesta de este trabajo a partir de una imagen en 2 dimensiones. En el Capítulo 5 se muestran 3 grupos de experimentos: el primer grupo, muestra las mallas generadas en 2 y 3 dimensiones con distintos tipos de imágenes; el segundo grupo de experimentos, analizan los parámetros requeridos por nuestro método y el tercer grupo de experimentos, analizan los tiempos de ejecución de los métodos propuestos en forma secuencial y paralela. Finalmente, el Capítulo 6, discute y expone las conclusiones de este trabajo, así como los trabajos futuros a partir de nuestra investigación.

Capítulo 2

Trabajos Relacionados

En este Capítulo, se realiza un resumen de los métodos que existen para construir mallas, las cuales pueden ser utilizadas para representar imágenes o simular fenómenos físicos y biológicos. De igual manera, como se ha explicado en el Capítulo 1, a diferencia de otros métodos de construcción de mallas, nuestro trabajo pretende utilizar la forma del objeto como parte del proceso de construcción de mallas. Para cumplir con este propósito, en este Capítulo presentamos un estudio de los métodos de esqueletización a partir de imágenes. Por otro lado, a diferencia de los métodos tradicionales de construcción de mallas, pretendemos insertar puntos sobre la imagen de manera regular y adaptativa de tal forma que puedan ser utilizados como generadores en una representación geométrica, como la triangulación de Delaunay. Para realizar esta distribución de puntos sobre la imagen estudiamos los métodos de muestreo de Poisson.

El resto de este Capítulo está organizado en 4 partes. En primer lugar, en la Sección 2.1 se presentan trabajos de la literatura para generar mallas. Después, en la Sección 2.2, se muestran los métodos para obtener la transformada de distancia a partir de una imagen. Luego, en la Sección 2.3 se examinan los trabajos para encontrar el esqueleto de una imagen. Al final de este Capítulo, la Sección 2.4 explica las técnicas para distribuir puntos, a través de discos de Poisson.

2.1. Generación de mallas

El problema de generación de mallas consiste en dividir un espacio en piezas simples llamadas elementos. De acuerdo a la forma de estos elementos, algunos trabajos pueden construir mallas triangulares (Qi et al., 2013), tetraedrales (Si, 2015), cuadrilaterales (Bommes et al., 2013), y hexaedrales (Kremer et al., 2014). Además, las mallas deben cumplir ciertas propiedades como: ajustarse a la forma del objeto, tener elementos que no sean muy grandes ni muy numerosos y deben estar compuestas por buenos elementos, donde un buen elemento es considerado aquel que es equilátero y equiangular, mientras que un mal elemento es aquel que es delgado y largo, parecido a la forma de una aguja.

De acuerdo a la estructura y organización de los elementos, existen mallas no estructuradas y estructuradas. Los trabajos propuestos por De Santis et al. (2010) y Ma et al. (2006) construyen mallas estructuradas, las cuales presentan vértices que pueden ser enumerados, de tal forma que se puede determinar qué vértices comparten un elemento a través de operaciones aritméticas. Por otro lado, algunos trabajos como los propuestos por Ibanez et al. (2016) y Smolarkiewicz et al. (2013) construyen mallas no estructuradas, las cuales almacenan explícitamente cada vértice junto con sus elementos y vértices vecinos.

En este trabajo, pretendemos proponer un algoritmo de construcción de mallas no estructuradas. Para generar este tipo de modelos, los trabajos propuestos por Schöberl (1997), Staten et al. (2010) y Löhner (2013) utilizan el método de avance frontal (Lo, 2013), el cual construye un elemento a la vez, comenzando por la frontera del dominio y avanzando hacia el interior y al exterior de la imagen. Los trabajos propuestos por Shephard y Georges (1991) y Ito et al. (2009) utilizan métodos basados en *grids* (Camata y Coutinho, 2013), los cuales generalmente utilizan las estructuras *quadtree* y *octree*, dependiendo de la dimensión, para subdividir el espacio e insertar vértices en la malla. Finalmente, los trabajos propuestos por Cuadros-Vargas et al. (2009), Si (2015) y Jamin et al. (2015) utilizan un enfoque basado en la triangulación de Delaunay (Frey y George, 2013).

Los métodos que construyen mallas a partir de una triangulación de Delaunay, a diferencia de los otros enfoques, pueden asegurar buenos elementos que cumplan con ciertas propiedades matemáticas usando un proceso de refinamiento. Por ejemplo, en 2 dimensiones, un refinamiento con propiedades de Delaunay, asegura que el mínimo ángulo en un elemento triangular es de 37 grados.

Entre los métodos basados en una triangulación de Delaunay, el método propuesto por Cuadros-Vargas et al. (2009), denominado Imesh, propone un algoritmo de construcción de mallas a partir de imágenes no preprocesadas. Además, a diferencia de los otros métodos en este grupo, el método de Cuadros-Vargas et al. (2009) construye un conjunto de submallas, donde cada submalla contiene una partición de la imagen original, la cual es obtenida en base a diferentes criterios como color y distancia. En este trabajo pretendemos conservar estas propiedades del método Imesh. Sin embargo, durante el proceso de construcción de la malla el método Imesh considera sólo la información del borde. A diferencia de este método, buscamos incluir información de la forma del objeto como parte del proceso de construcción de la malla. Para lograr este objetivo, a continuación en la siguiente Sección, presentamos un estudio de los métodos que extraen información de la distancia entre elementos a través de la transformada de distancia.

2.2. Transformada de distancia

La transformada o mapa de distancia es una representación ampliamente utilizada en procesamiento de imágenes (Xue et al., 2012), visión computacional (Al-Kofahi et al., 2010) y reconocimiento de patrones (Tani et al., 2016). Esta representación describe una imagen binaria a través de las distancias mínimas hacia una región de interés. Estas dis-

tancias en una imagen pueden ser calculadas, usando distintas métricas como la euclideana (Mishchenko, 2015; Man et al., 2010), chamfer (Tran, 2013) o Manhattan (Bailey, 2012).

De acuerdo a la forma en cómo las distancias se propagan a través de la imagen, los trabajos pueden ser clasificados en 2 grupos, descritos a continuación:

2.2.1. **Métodos basados en frentes de onda**. Uno de los primeros métodos en este grupo es el trabajo propuesto por Sethian (1996), denominado *Fast Marching Method* (FMM), el cual propone un algoritmo secuencial de orden $O(nlog(n))$. Este método y algunas variaciones (Gomez et al., 2015), expanden en cada iteración un subconjunto de puntos denominado frente de propagación, hasta cubrir toda la imagen, donde cada punto procesado, aproxima la ecuación Eikonal (Wong y Leung, 2016) para estimar su distancia. Telea y Van Wijk (2002) proponen una versión modificada de FMM denominada *Augmented Fast Marching Method*, en donde además de las distancias, cada elemento de la imagen, guarda el punto del frente de propagación inicial del cual se ha originado. La razón por la que FMM es de orden $O(nlog(n))$, se debe a que mantiene una estructura *heap*, la cual se actualiza cada vez que un elemento es procesado. Luego, basado en este modelo, el trabajo propuesto por Yatziv et al. (2006) reemplaza la estructura heap por un arreglo de listas enlazadas, consiguiendo un método secuencial de orden $O(n)$. Más adelante, Jeong y Whitaker (2008) propusieron una implementación paralela de FMM denominada *Fast Iterative Method* (FIM), usando threads de la Unidad de Procesamiento Central o *Central Processing Unit* (CPU). En este método, cada elemento que pertenece al actual frente de propagación, es procesado al mismo tiempo. En esta línea, y basado en el desarrollo de tarjetas gráficas, trabajos como Yang y Stern (2017), Yang y Stern (2015) y Dang et al. (2013) proponen una implementación paralela, utilizando la Unidad de Procesamiento Gráfico o *Graphic Processing Unit* (GPU), de los métodos FMM y FIM, respectivamente.

2.2.2. **Métodos basados en barridos**. A diferencia de los métodos basados en frentes de onda, este grupo de métodos, comienzan la propagación por un lado de la imagen hasta llegar al otro lado de la imagen, realizando varias pasadas en distintas direcciones. En este grupo de métodos, el trabajo propuesto por Meijster et al. (2002) plantea un escaneo de una imagen en 2 dimensiones realizado en 2 fases, donde cada fase procesa cada dirección de la imagen. Otro algoritmo, denominado *Fast Sweeping Method* (FSM), propuesto por Zhao (2004), procesa una imagen aproximando la ecuación de Godunov (Zhang et al., 2006) en cada punto, obteniendo un algoritmo de orden $O(n)$. Más adelante, tanto Zhao (2007) como Detrixhe et al. (2013) formulan una implementación paralela en CPU de FSM, donde cada dirección de la imagen es tratada simultáneamente. Por otro lado, Cao et al. (2010b) proponen una implementación paralela en GPU, la cual subdivide la imagen, en pequeños grupos, los cuales son procesados con una propagación por barridos y después son combinados en el resultado final. Un trabajo más reciente propuesto por Hong y Jeong (2016), procesa una imagen en 3 dimensiones, utilizando múltiples GPU los cuales procesan cada plano de la imagen al mismo tiempo.

Como se ha mencionado en esta Sección, la transformada de distancia de una imagen

binaria, es una representación, en donde a cada elemento se le asigna su mínima distancia hacia una región de interés. Los métodos basados en barridos generalmente son más rápidos que los métodos basados en frentes de onda, debido, en parte, a que no deben mantener una estructura, y también debido a que el escaneo de cada fila en cada dirección de la imagen puede ser paralelizado. En contraste, los métodos basados en frentes de onda, aunque deben mantener una estructura *heap*, pueden definir un número de iteraciones, sin tener la necesidad de procesar la imagen completa.

A continuación, en la Sección 2.3, se mostrarán algunos trabajos que se encargan de extraer el esqueleto de una imagen.

2.3. Esqueletización

La esqueletización es un proceso que genera una representación mínima de algún objeto en una imagen, considerando sus propiedades geométricas y topológicas (Saha et al., 2015). De acuerdo a su dimensión, existen 2 tipos de esqueleto: esqueleto superficial, el cual está conformado por la unión de estructuras en 1D y 2D y esqueleto curvo, compuesto sólo por estructuras 1D.

A lo largo de los años, muchos métodos han sido propuestos para obtener el esqueleto de un objeto a partir de una imagen. Estos métodos pueden ser agrupados en 4 categorías (Saha et al., 2016; Sobiecki et al., 2014; Saha et al., 2017), descritas a continuación:

2.3.1. **Métodos basados en campos de distancia**. Este grupo de métodos modelan el principio de evolución de curvas, donde el esqueleto está conformado por las singularidades de una transformada de distancia, es decir, los puntos de colisión entre 2 frentes opuestos. Leymarie y Levine (1992) modelaron la evolución de las curvas utilizando *active contours* para una frontera en 2 dimensiones. El trabajo propuesto por Kimmel et al. (1995), utiliza un modelo basado en *level sets* a partir de un mapa de distancia para localizar los puntos que conforman el esqueleto en 2 dimensiones. Otros trabajos utilizan el mapa de distancia para realizar un proceso de selección e identificar puntos que potencialmente pueden formar parte del esqueleto. Ge y Fitzpatrick (1996) realizan este proceso de selección detectando discos máximos, Pudney (1998) utiliza una operación morfológica de *thinning*, Bouix y Siddiqi (2000) utilizan el cálculo de la divergencia y Bitter et al. (2001) emplean la gradiente. Más adelante, Siddiqi et al. (2002), Bouix et al. (2005) y Siddiqi y Pizer (2008), modelan la evolución de curvas a través de la ecuación *Hamilton-Jacobi*, donde el esqueleto se ubica en las singularidades del flujo exterior del campo de vectores, generado por la gradiente de la transformada de distancia. Esta ecuación puede ser extendida a 3 dimensiones y puede ser calculada de forma paralela. Otro algoritmo que puede ser implementado de forma paralela en base a un mapa de distancia euclidiano, es el propuesto por Hesselink y Roerdink (2008), el cual permite obtener esqueletos a partir de imágenes en 2 y 3 dimensiones. Trabajos más recientes como el propuesto por Gao et al. (2018), obtienen el esqueleto a partir de una imagen binaria a través

de la ecuación de calor, sin la necesidad de definir parámetros de entrada definidos por el usuario.

2.3.2. **Métodos geométricos**. En este grupo de métodos se utilizan propiedades geométricas para formar un esqueleto. Los trabajos propuestos por Brandt y Algazi (1992) y Ogniewicz y Ilg (1992), generan un diagrama de Voronoi (Edelsbrunner, 2014) a partir de un conjunto de puntos sobre la frontera de una imagen en 2 dimensiones. De esta forma, el esqueleto está formado por las aristas internas del diagrama de Voronoi generado. Schmitt (1989) demostró que mientras el número de puntos sea mayor, este tipo de métodos converge hacia un esqueleto continuo. Más adelante, los trabajos propuestos por Turkiyyah et al. (2000), Hisada et al. (2001) y Dey y Zhao (2004) extendieron la extracción del esqueleto a partir del diagrama de Voronoi para 3 dimensiones. Por otro lado, el trabajo de Amenta et al. (2001) utiliza los centros de un conjunto de discos inscritos en el interior y exterior de la frontera de una imagen para obtener el esqueleto en 2 dimensiones. Los trabajos propuestos por Au et al. (2008) y Cao et al. (2010a), están basados en un conjunto de puntos en 3 dimensiones, que son usados para construir una malla, la cual se va contrayendo en cada iteración hasta formar un esqueleto delgado. Otro método que utiliza una representación de mallas poligonales para obtener el esqueleto es el propuesto por Jalba et al. (2013), el cual desarrolló un método paralelo en GPU para extraer esqueletos curvos y superficiales en 3 dimensiones.

2.3.3. **Métodos basados en adelgazamiento**. Este grupo de métodos recurre a la erosión de los elementos de la imagen que pueden ser eliminados sin alterar la topología del objeto. Para detectar los elementos que pueden ser eliminados, en cada iteración se puede utilizar distintos *kernels* o subiteraciones. Así por ejemplo, Palàgyi y Kuba (1998) usan 6 subiterationes, mientras que Palágyi y Kuba (1999) utilizan 12 subiteraciones para adelgazar una imagen hasta obtener su esqueleto. La detección de estos elementos se puede realizar de manera local, por este motivo, Zhang y Suen (1984), Lei Huang et al. (2003) y Kwon (2013), propusieron métodos de adelgazamiento de forma paralela. Sin embargo, para poder adelgazar una imagen, se debe definir un conjunto de *kernels* que ayuden a identificar los elementos que pueden ser eliminados. Especialmente en 3 dimensiones, se tiene una gran cantidad de posibles combinaciones. Por esta razón, (Homann, 2007) plantea el uso de árboles de decisión para cubrir las combinaciones posibles. Por otro lado, Kaur y Sharma (2013) proponen un proceso de adelgazamiento que puede ser guiado utilizando la gradiente de la imagen y los trabajos de Chang et al. (2013) y Chang et al. (2014) se orientan a través de la transformada de distancia. Trabajos más recientes como el propuesto por Jin y Kim (2017) utilizan 2 conjuntos de *kernels*, uno para erosionar la imagen y otros para corregir los posibles elementos desconectados.

2.3.4. **Métodos basados en campos generales**. Los métodos en este grupo por lo general convierten una imagen a una representación de vectores, la cual se analiza para obtener el esqueleto de la imagen. Los trabajos propuestos por Ahuja y Jen-Hui Chuang (1997), Min-Chi Ko et al. (2000) y Cornea et al. (2005) representan la imagen a través de campos de potencial, donde cada elemento que conforma la frontera de la imagen actúa como una carga puntual positiva y el resto de elementos

calculan la sumatoria de potencial hacia esas cargas. Esta representación puede ser generada de forma paralela en GPU como lo proponen Lu y Xuewen (2014). A partir de la imagen como un campo de potencial, se identifican los puntos críticos y se busca un camino entre ellos para conectarlos y obtener el esqueleto de la imagen (Abdel-Hamid y Yee-Hong Yang, 1994). De forma similar, Hassouna y Farag (2009) representa una imagen a través del flujo de vector de gradiente o GVF para obtener el esqueleto de una imagen en 3 dimensiones.

Como se ha mencionado en esta Sección, existen diferentes métodos que extraen el esqueleto a partir de una imagen. Los métodos geométricos, por lo general construyen un diagrama de Voronoi usando un conjunto de puntos sobre la frontera de la imagen. Sin embargo, al tratarse de un enfoque geométrico, aumentan su complejidad al tratar imágenes en 3 dimensiones. Los métodos basados en adelgazamiento, pueden ser implementados de forma paralela pero aún así, están basados en un conjunto de iteraciones. Por otro lado, tanto los métodos basados en campos de distancia como los métodos basados en campos generales, convierten la imagen en una representación que pueda ser analizada para obtener el esqueleto de la imagen. En ambos casos, la representación puede ser generada de forma paralela, pero, el análisis en los métodos basados en campos generales se realiza de forma iterativa. En el caso de los métodos basados en campos de distancia, el esqueleto puede ser extraído de forma paralela analizando, por ejemplo, el flujo exterior a partir de la gradiente de un mapa de distancia generado por la frontera de una imagen.

A continuación, en la Sección 2.4, se mostrará algunos algoritmos que generan muestras de manera uniforme y adaptativa utilizando discos de Poisson.

2.4. Muestreo de discos de Poisson

Una de las aplicaciones más importantes de la distribución de puntos en computación gráfica es el muestreo. Como se menciona en el trabajo propuesto por Yan et al. (2015), las distribuciones de puntos pueden ser clasificadas observando el espectro generado a través de Fourier. Diferentes espectros están asociados a distintos colores. Por ejemplo, el ruido blanco, presenta un espectro plano y es generalmente usado en la generación de números aleatorios; y el ruido azul (Chen et al., 2012; Li et al., 2010) que presenta distribuciones de poca energía con baja frecuencia, genera muestras uniformes aleatorias.

Esta investigación se enfoca en los trabajos que generan distribuciones de puntos con propiedades de ruido azul, a través de discos de Poisson. Esta distribución de puntos puede ser realizada de 2 formas, de manera uniforme, donde las muestras se separan como mínimo por una distancia constante o de manera adaptativa, en base a una función que define la distancia entre muestras. Los trabajos para generar estas distribuciones pueden ser divididos en 2 grupos, descritos a continuación:

2.4.1. **Muestreo de discos de Poisson**. Entre los primeros algoritmos con la capacidad de producir muestras adaptativas se encuentra el propuesto por Cook (1986), de-

nominado *Dart Throwing*. En cada iteración este algoritmo evalúa la vecindad de un nuevo punto aleatorio, para asegurarse que no haya ningún conflicto con alguna muestra aceptada. Durante este proceso iterativo es difícil precisar el número de iteraciones, así como, el número total de muestras. Por esta razón, los trabajos de McCool y Fiume (1992) y Yuksel (2015) proponen métodos basados en obtener un determinado número de muestras. Además, para mejorar el análisis de vecinos, Bridson (2007), Dunbar y Humphreys (2006) y White et al. (2007) aprovechan estructuras de datos espaciales como *grids* y *heaps* para organizar la búsqueda de vecinos. Más adelante, Wei (2008) abrió el camino proponiendo un método adaptativo y paralelo en GPU. Este método, genera una estructura *grid* sobre la imagen y genera muestras en cada cuadrante, evitando de esta forma algún conflicto entre las muestras. Otro método paralelo en GPU que utiliza una estructura *grid*, es el propuesto por Bowers et al. (2010), donde la estructura *grid* se agrupa en *phase groups* y los cuadrantes que pertenecen a un mismo *phase group* se procesan al mismo tiempo. Por otro lado, el trabajo de Ying et al. (2013), asigna una prioridad única para cada posible muestra, la cual se utiliza para resolver conflictos. Este método utiliza una nube de puntos y puede ser implementado de forma paralela en CPU y GPU.

2.4.2. **Muestreo de discos de Poisson maximal**. Aunque el muestreo de discos de Poisson genera muestras aleatorias uniformes y adaptativas, este no garantiza una distribución maximal, donde cada espacio posible es ocupado por alguna muestra. Jones (2006) presenta el primer método para generar una distribución maximal, utilizando el diagrama de Voronoi para identificar las regiones que no han sido ocupadas por alguna muestra. De forma similar, el trabajo de Gamito y Maddock (2009) recurre a una estructura *grid* para encontrar las regiones vacías. Ebeida et al. (2011) por su parte, proponen un algoritmo de dos fases que puede ser implementado de forma paralela en GPU, estas fases están basadas en el método *Dart Throwing* y en el particionamiento del espacio. Más adelante, el trabajo de Ebeida et al. (2012) usa una estructura *quadtree*, la cual requiere menos memoria para detectar y cubrir las regiones vacías. Los trabajos que han sido mencionados anteriormente generan distribuciones maximales uniformes, pero el trabajo de Yan y Wonka (2013) formula un método para generar distribuciones maximales y adaptativas de acuerdo a una función definida por el usuario. Por otro lado, Ying et al. (2014) propone un trabajo que recibe una distribución de puntos aleatoria y la transforma en una distribución maximal.

En esta Sección se han mencionado los métodos para generar muestras utilizando discos de Poisson. Las distribuciones maximales utilizan por lo general alguna estructura de datos adicional que permita buscar e identificar las regiones que no están siendo cubiertas por alguna muestra. Durante la investigación de este trabajo no se ha encontrado algún método que proponga generar distribuciones maximales, de forma adaptativa y paralela en GPU. Por otro lado, las distribuciones no maximales, por lo general dividen el espacio para poder generar muestras de forma independiente en cada cuadrante y proponer métodos de forma paralela.

En este Capítulo se ha elaborado un estudio de los diversos métodos para generar un mapa de distancia y un esqueleto a partir de una imagen, así como, para obtener muestras

usando discos de Poisson. Tomando en cuenta estos temas, más adelante, en el Capítulo 3 se detalla la implementación de los algoritmos que forman parte de la propuesta de este trabajo, descrita en el Capítulo 4.

Capítulo 3

Marco Teórico

En el Capítulo anterior se han mencionado algunos métodos para encontrar la transformada de distancia y el esqueleto de una imagen, así como los métodos para distribuir puntos utilizando discos de Poisson. Este Capítulo se enfoca en describir los métodos utilizados en nuestra propuesta. En primer lugar, en la Sección 3.1, se describe el trabajo de Cuadros-Vargas et al. (2009), para construir mallas a partir de imágenes, ya que nuestra propuesta en el Capítulo 4, reformula este trabajo. Más adelante, en la Sección 3.2, se presenta el algoritmo propuesto por Bouix et al. (2005) para extraer un esqueleto de una imagen a partir de un mapa de distancias. Después, en la Sección 3.3, se detalla el trabajo de Ying et al. (2013) para distribuir un conjunto de puntos usando discos de Poisson.

3.1. Algoritmo *Imesh*

El algoritmo *Imesh* propuesto por (Cuadros-Vargas et al., 2009), es un método que construye mallas simpliciales a partir de imágenes no preprocesadas en 2 y 3 dimensiones. Este algoritmo es utilizado como punto de partida para proponer un nuevo método, el cual está detallado en el Capítulo 4. Como se ha mencionado en el Capítulo 1, el método *Imesh* está dividido en 3 etapas: Construcción ($Imesh_{Cm}$), Particionamiento ($Imesh_{Pm}$) y Mejoramiento ($Imesh_{Mm}$). Para las intenciones de este trabajo, analizaremos la etapa $Imesh_{Cm}$.

En la descripción de $Imesh_{Cm}$, se utilizará el término d para referirnos a elementos en 2 y 3 dimensiones, de la siguiente manera. La palabra $celula_d$ será utilizada para indicar triángulo o tetraedro. La palabra $face_d$ se usará para referir a la arista de intersección entre 2 triángulos o al triángulo de intersección entre 2 tetraedros. El término $area_d$ será utilizado para expresar área de un triángulo, o volumen de un tetraedro, y la expresión $pixel_d$, se usa para referir a los componentes de la imagen, *pixel* o *voxel*.

3.1.1. Construcción de la malla ($Imesh_{Cm}$)

Una imagen, representada por el símbolo $\mathcal{IMG}$, es considerada como una función definida en un dominio $D \subset N^d$ de la forma $D = [0, ..., N_1]$ x ... x $[0, ..., N_d]$, $d \in \{2, 3\}$, donde N es el conjunto de los números naturales. La función $\mathcal{IMG}$ asigna a cada elemento del dominio D un número real positivo, $\mathcal{IMG}$: $D \rightarrow R^+$. Cada par $(u,\mathcal{IMG}(u)) \in D$ x R^+ es denominado *pixel* o *voxel*, dependiendo si $D \subset N^2$ o $D \subset N^3$, respectivamente.

Dada una imagen $\mathcal{IMG} : D \rightarrow R^+$, la etapa de construcción de la malla busca seleccionar un conjunto de puntos $C \subset D$ tal que las $celulas_d$ contenidas en la triangulación de Delaunay (D_T) de C capturen, de alguna forma, características de la imagen. Por ejemplo, se puede buscar que las $celulas_d$ en D_T estén contenidas en las regiones de D donde $\mathcal{IMG}$no varía bruscamente.

Sea $\mathcal{IMG} : D \rightarrow R^+$ una imagen y D_T la triangulación de Delaunay generada a partir de un conjunto de puntos C_0 en la frontera de D, o sea, cada punto en C_0 es de la forma $u = (u_1, ..., u_d)$, $d \in \{2, 3\}$, donde por lo menos una de las coordenadas u_j es igual a cero o igual a N_j, para algún $j = 1, ..., d$.

Considerando T como conjunto de $celulas_d$ en D_T y $E : T \rightarrow R^+$ una función que asocia, a cada $celula_d$ $\sigma \in T$, una medida de error. El objetivo es que la función E mida, si dada una $celula_d$ σ se ajusta a una determinada propiedad, posibilitando decidir si σ debe o no permanecer a la triangulación.

Algunas técnicas de modelamiento de imágenes, en general, definen la función E como una medida de error de interpolación. La mayoría de estos métodos evalúan la función E recorriendo, para cada $celula_d$ σ, los elementos de la imagen contenidos en el interior de σ. Cuando E señala a σ como una $celula_d$ no apropiada, la triangulación es actualizada para eliminar σ. Tal actualización es, generalmente, llevada a cabo por medio de la inserción de nuevos vértices. Aunque esta estrategia es muy utilizada, presenta dos inconvenientes principales. El primer problema, es que la iteración por el interior de las $celulas_d$ puede exigir un gran esfuerzo computacional, porque para cada actualización de la triangulación, todas las nuevas $celulas_d$ generadas deben volverse a visitar con el fin de evaluar E. El segundo problema, es que la inserción de nuevos vértices puede provocar una acumulación de puntos, en regiones particulares del dominio, introduciendo un número excesivo de elementos en la triangulación.

Con el objetivo de evitar los problemas mencionados anteriormente, *Imesh* define una función de error E, basada en el recorrido por las medianas de las $celulas_d$. Además, E considera la posición de los vértices existentes en la triangulación, procurando evitar la acumulación de puntos. De esa forma, la función de error propuesta reduce el esfuerzo computacional, produciendo triangulaciones con buenas propiedades.

Específicamente, sean h_0,...,h_d las $d + 1$ medianas de una $celulas_d$. Considerando el conjunto de puntos $U^{h_j} = \{u \in h_j \mid \mathscr{E}(u) \geq c_E\}$, donde $\mathscr{E}$ es una medida de error puntual en u y c_E es un escalar positivo. El conjunto U^{h_j} contiene los puntos sobre la mediana h_j cuyo error puntual $\mathscr{E}$ es mayor que un valor deseado, definido por c_E.

$Imesh$ denota con el símbolo $d_{D_T}(u)$ al cuadrado de la distancia de u al vértice más próximo en D_T, es decir, $d_{D_T}(u) = min\{d_e^2(u, v_s)\}$, donde $d_e(\cdot, \cdot)$ es la distancia euclidiana y v_s un vértice de D_T, sea u_{h_j} el punto de U^{h_j} donde d_{D_T} es máximo. A partir de las definiciones anteriores, $Imesh$ define la función de error E según la Ecuación 3.1.

$$E(\sigma) = \begin{cases} max\{\mathscr{E}(u_{h_j})\} & \text{si } U^{h_j} \neq \emptyset \text{ para algún } j = 0, ..., d \\ 0 & \text{caso contrario} \end{cases} \tag{3.1}$$

De esta forma, la función E, considera entre los puntos cuyo error puntual está encima de un umbral, el más lejano, en cada mediana, de los vértices de la triangulación. Entre los más lejanos, E asume el valor del mayor error puntual. Así, el punto u_{h_j} escogido, es decir, el punto tal que $\mathrm{E}(\sigma) = \mathscr{E}(u_{h_j})$, es insertado en la triangulación.

Las dos operaciones principales de la función de error E son el cálculo del error puntual $\mathscr{E}$ y el cálculo del punto más distante, las cuales se detallan a continuación.

3.1.1.1. Cálculo del error puntual

Existen varias posibilidades para la definición de la función de error puntual, dado que diferentes definiciones de $\mathscr{E}$ originan diferentes conjuntos de puntos y por lo tanto, diferentes triangulaciones. $Imesh$ presenta 2 definiciones distintas para la función $\mathscr{E}$, siendo cada una de ellas más apropiada para un determinado tipo de aplicación.

- $\mathscr{E}$ **como error de interpolación lineal**. Una forma inmediata de definir la función de error puntual es considerar el error de interpolación linear. Formalmente, se considera $\sigma = [v_0, ..., v_d]$ una $celula_d$ y $\alpha_i, i = 0, ..., d$ las coordenadas baricéntricas de un punto $u \in h_j$, siendo h_j una de las medianas de σ. Sea $\tilde{u} = \alpha_0 \mathcal{IMG}(v_0) + ... + \alpha_d \mathcal{IMG}(v_d)$ la interpolación en u de los valores de $\mathcal{IMG}$ en los vértices de σ. Por lo tanto, se puede definir la Ecuación 3.2.

$$\mathscr{E}_I(u) = |\tilde{u} - \mathcal{IMG}(u)| \tag{3.2}$$

  Utilizando a $\mathscr{E}_{\mathcal{IMG}}(u)$ como el error puntual, la función de error $\mathscr{E}$ busca localizar $celulas_d$ en regiones donde la aproximación lineal por partes de $\mathcal{IMG}$ no es buena, considerando la densidad de vértices.

- $\mathscr{E}$ **como medida de homogeneidad**. Cuando el objetivo es crear mallas apropiadas para distinguir objetos en ellas, o sea, para ser segmentadas, se debe intentar garantizar que las $celulas_d$ de la triangulación estén contenidas en regiones homogéneas de la imagen. Es decir, una misma $celula_d$ no debe atravesar objetos distintos de la imagen. Pero $\mathscr{E}_{\mathcal{IMG}}(u)$ no asegura tal propiedad. Para cumplir esta propiedad, $Imesh$ define una función de error puntual que busca nuevos vértices en la frontera del objeto en la imagen. Esta función puede ser declarada de la siguiente manera:

Sea $A(u) = min\{\alpha i\}, i = 0, ..., d$, una función que asocia a cada punto u, de las medianas de una $celula_d$ σ, el $area_d$ de su menor coordenada baricéntrica. Considerando $H(u)$ una función binaria capaz de determinar si un punto u corresponde, o no, a una coordenada del borde en la imagen $\mathcal{IMG}$. El método *Imesh* define la función $H(u)$ de 2 formas distintas:

- **Como un filtro que detecta altas frecuencias en una imagen**. Sea $G(u)$ un extractor de borde, como: Sobel, Prewitt, Roberts, o Canny; y c_g un umbral que representa los valores de alta frecuencia a ser considerados, se define la Ecuación 3.3.

$$H_g(u) = \begin{cases} 0 & \text{si } G(u) < C_g \\ 1 & \text{si } G(u) \geq C_g \end{cases} \tag{3.3}$$

- **Como una función que define fronteras basadas en un conjunto de isovalores**. Sea $\psi = \{v_0, v_1, ..., v_n\}$, un conjunto de isovalores que definen el borde de regiones contenidas en la imagen $\mathcal{IMG}$ y U_{ant}, el punto anterior al punto u durante el recorrido sobre una mediana de una $celula_d$, se define la Ecuación 3.4.

$$H_v(u) = \begin{cases} 1 & \text{si } \mathcal{IMG}(u) \in \psi \\ 1 & \text{si } \exists v_i \in \psi, v_i \in]\mathcal{IMG}(u), \mathcal{IMG}(U_{ant})[\\ 0 & \text{caso contrario} \end{cases} \tag{3.4}$$

A partir de las definiciones anteriores, *Imesh* establece las medidas de homogeneidad $\mathscr{E}_g(u)$ y $\mathscr{E}_v(u)$, basadas en los conceptos de altas frecuencias y conjunto de isovalores según la Ecuación 3.5 y la Ecuación 3.6, respectivamente.

$$\mathscr{E}_g(u) = A(u)H_g(u) \tag{3.5}$$

$$\mathscr{E}_v(u) = A(u)H_v(u) \tag{3.6}$$

Las funciones $\mathscr{E}_g$ y $\mathscr{E}_v$ calculan, para cada punto del borde en las medianas, el valor de la $area_d$ asociada a su menor coordenada baricéntrica. Siendo $u \in h_j$ un punto supuestamente en el borde de objetos de la imagen, las funciones $\mathscr{E}_g$ y $\mathscr{E}_v$, proporcionan una medida de cuánto la $celula_d$ σ invade un objeto de la imagen, o sea, valores pequeños de $\mathscr{E}_g$ o $\mathscr{E}_v$ significan que h_j intercepta el borde de un objeto de la imagen próximo a una $face_d$ de la $celula_d$. Esto se puede observar en la Figura 3.1.

Consecuentemente, cuando $\mathscr{E}_g$ y $\mathscr{E}_v$ son utilizadas, la función E, estará midiendo si dada una $celula_d$ σ está, o no, bien "encajada" en una región de la imagen. Por lo tanto, valores pequeños de E indican $celulas_d$ bien posicionadas.

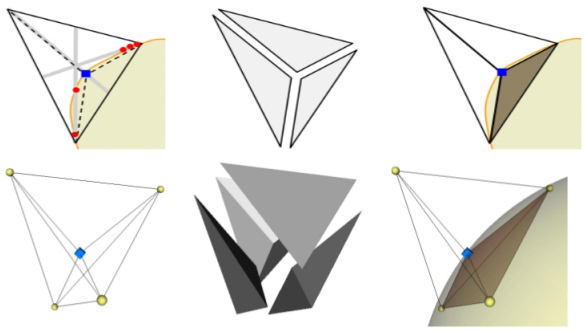


Figura 3.1: (Cuadros-Vargas et al., 2009). Medida de error puntual de homogeneidad. En 2 dimensiones (primera fila), los círculos representan el conjunto U^{h_j} de puntos seleccionados en las medianas de una $celula_d$. El cuadrado representa el punto escogido de este conjunto con el cual el error de la $celula_d$ es calculado. Luego, la idea es extendida directamente para 3 dimensiones (segunda fila).

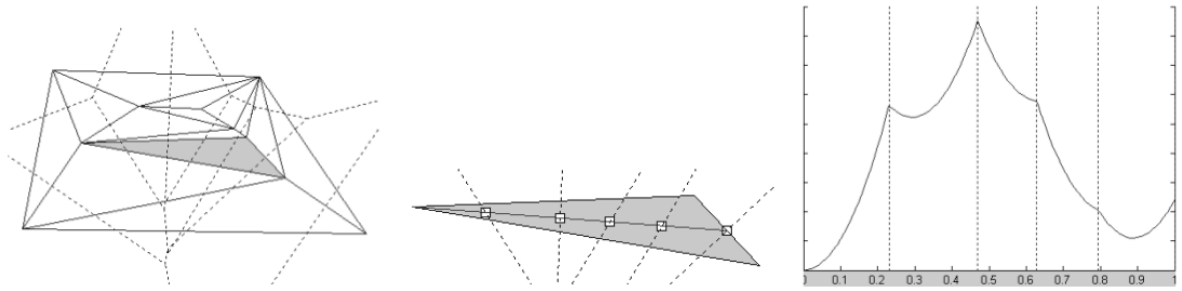
3.1.1.2. Cálculo del punto más distante de los vértices de la triangulación

El costo computacional involucrado en el cálculo de los puntos u_{h_j} puede imposibilitar el uso práctico de $Imesh$. Por otro lado, la Proposición 3.1.1 descrita abajo, asegura que tales puntos pueden ser calculados de forma eficiente.

Proposición 3.1.1 *Si una mediana h_j intercepta m células del diagrama de Voronoi entonces el punto u_{h_j} puede ser calculado como máximo en $2m-1$ cálculos de distancia.*

Prueba. Sean v y x los extremos de h_j, donde v es un vértice de la $celula_d$ σ (en la triangulación) y x es el centroide de la $face_d$ opuesta a v en σ. Suponiendo que h_j está parametrizada de v a x. Por hipótesis, la mediana h_j intercepta m células del diagrama de Voronoi. De esta forma, los puntos en U^{h_j} pueden ser divididos en subconjuntos $U_i^{h_j}$, conteniendo cada uno de los puntos de U^{h_j} que están en una de las m células del diagrama de Voronoi. Como se ilustra en la Figura 3.2 (para el caso bidimensional), la función d_{D_T} es cuadrática por partes en h_j. De este modo, en cada subconjunto $U_i^{h_j}$ el máximo de d_{D_T} ocurre o en el punto más a la izquierda o en el punto más a la derecha de $U_i^{h_j}$ (considerando la parametrización), o sea, sólo 2 de los cálculos de distancia son suficientes para encontrar el máximo d_{D_T} en cada subconjunto. Un caso particular ocurre en el subconjunto $U_0^{h_j}$, el cual contiene los puntos de la célula de Voronoi asociada a v. Como d_{D_T} es igual a cero en v (ver Figura 3.2c), d_{D_T} necesita ser evaluada solamente en el punto más a la derecha de $U_0^{h_j}$.

Aunque la Proposición 3.1.1 proporciona un mecanismo eficiente para calcular u_{h_j}, los cálculos de intersección entre h_j y las células de Voronoi son inevitables, lo que puede perjudicar la robustez del algoritmo. Una alternativa que presenta buenos resultados prácticos es calcular u_{h_j} como el punto en U^{h_j} más cercano al circuncentro de σ. En las pruebas que $Imesh$ presenta, aproximadamente el 80 % de los puntos que maximizan d_{D_T} son también los más cercanos al circuncentro, es decir, ambos enfoques son equivalentes en aproximadamente 80 % de los casos. La Figura 3.3 ilustra este hecho, la Figura 3.3a presenta la imagen original; la Figura 3.3b y la Figura 3.3c, presentan una comparación entre dos mallas generadas con la estrategia de mayor distancia y utilizando el punto cercano al circuncentro, respectivamente. En ambos casos se empleó una función de error puntual



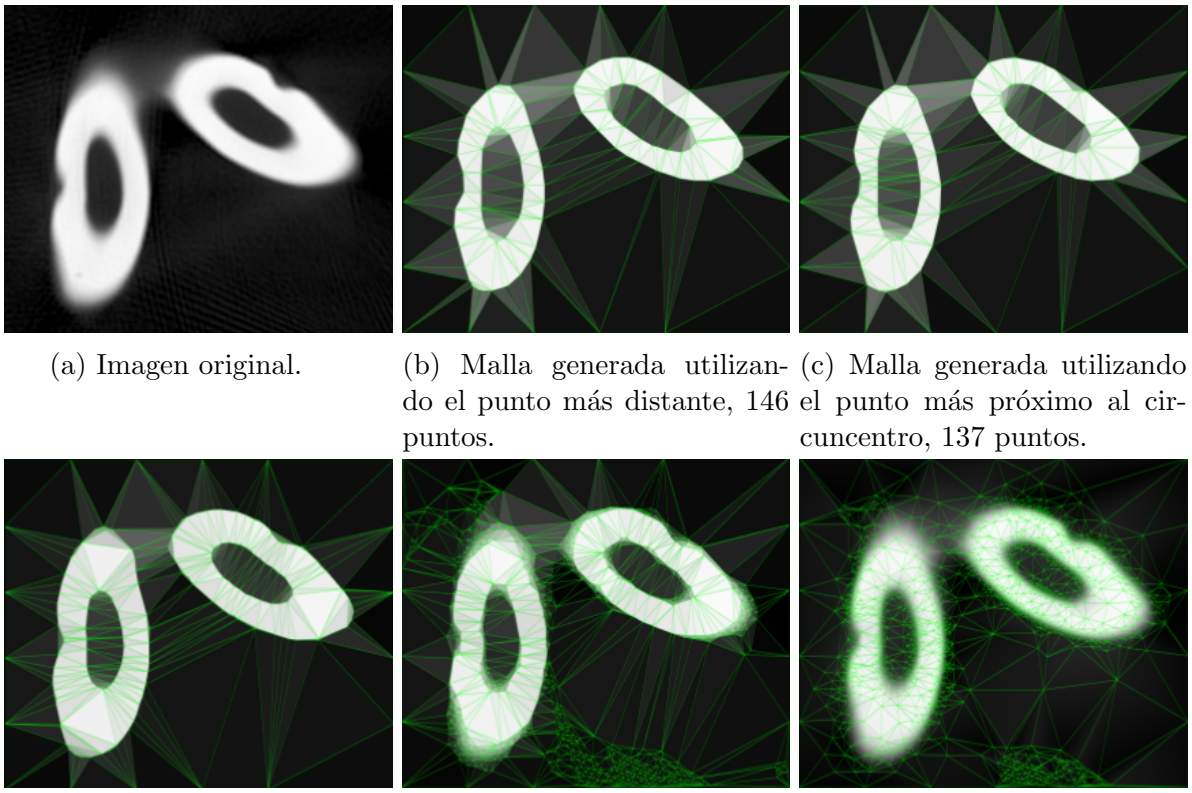
(a) Triangulación de Delaunay y diagrama de Voronoi.

(b) Mediana de un triángulo interceptando las células de Voronoi

(c) d_{D_T} es cuadrática por partes en h_j.

Figura 3.2: (Cuadros-Vargas et al., 2009). Comportamiento de la función d_{D_T}.

de homogeneidad. La Figura 3.3d, la Figura 3.3e y la Figura 3.3f, muestran resultados obtenidos utilizando las medidas de error $\mathscr{E}_v$, $\mathscr{E}_g$ y $\mathscr{E}_i$, respectivamente.



(a) Imagen original.

(b) Malla generada utilizando el punto más distante, 146 puntos.

(c) Malla generada utilizando el punto más próximo al circuncentro, 137 puntos.

(d) Error como medida de homogeneidad $\mathscr{E}_v$, 225 puntos.

(e) Error como medida de homogeneidad $\mathscr{E}_g$, 1070 puntos.

(f) Error de interpolación linear $\mathscr{E}_i$, 1936 puntos.

Figura 3.3: (Cuadros-Vargas et al., 2009). Comparaciones entre los criterios de cálculos de distancia, Figuras (b) y (c); y comparaciones entre cálculos de error puntual, Figuras (d) (e) y (f).

En esta Sección, se ha descrito la etapa de Construcción del método *Imesh*, la cual es utilizada para producir nuestra propuesta. En base a este análisis, nuestro trabajo, considera no sólo la información del borde de la imagen, como lo hace el método *Imesh*,

sino que también planteamos utilizar información sobre su forma, para producir una malla que se adapte mejor a los detalles del borde. Para lograr esto, en la siguiente Sección, se define el método de esqueletización que va a ser utilizado más adelante en nuestra propuesta en el Capítulo 4.

3.2. Esqueletización

La esqueletización es un proceso que genera una representación compacta de un objeto, reduciendo su dimensión, al mismo tiempo, conservando sus propiedades geométricas y topológicas.

Como se ha mencionado en la Sección 2.3, existen distintos métodos para obtener esta representación a partir de imágenes. Ya que uno de los objetivos en este trabajo es proponer un método paralelo extensible de 2 a 3 dimensiones, los métodos basados en geometría (Cao et al., 2010a; Jalba et al., 2013) fueron descartados, porque aumentan su complejidad en el contexto de 3 dimensiones. Por otro lado, los métodos basados en erosiones continuas, también fueron descartados, debido a que necesitan de un objeto ya segmentado y aunque una parte de su procedimiento puede ser paralelizado otra aún sigue siguen siendo iterativa. Los 2 grupos restantes, basados en campos generales (Lu y Xuewen, 2014; Min-Chi Ko et al., 2000) y campos de distancia (Gao et al., 2018; Bouix et al., 2005), ambos convierten la imagen en una representación de vectores o distancias, respectivamente. Sin embargo, aunque los métodos basados en campos generales pueden encontrar puntos críticos de forma paralela, necesitan de un proceso secuencial para conectar los puntos críticos que conforman el esqueleto. Por este motivo los métodos que siguen este enfoque, aunque con buenos resultados, son descartados para el propósito de nuestra investigación por no proponer procesos paralelos en cada una de sus etapas. Por otro lado, el método de Bouix et al. (2005), basado en campos de distancia, propone un algoritmo paralelo para obtener las discontinuidades en la gradiente, seguido de una etapa de *thining* para calcular el esqueleto. Esta última etapa aunque presenta un proceso iterativo, puede ser reemplazado por un método de umbralización, el cual puede ser implementado de forma paralela, como se explicará más adelante en el Capítulo 4. Por las razones anteriores, esta es la estrategia que utilizaremos en este trabajo de investigación.

El algoritmo original propuesto por Bouix et al. (2005), consta a grandes razgos de 3 etapas. Primero, calcula un mapa de distancia en base a una frontera. Segundo, con el mapa de distancia obtiene la gradiente y en base a esta gradiente genera un mapa de flujo. Por último, utilizando la información del mapa de flujo, realiza un proceso secuencial de *thinning* guiado por la gradiente y el mapa de distancia. De esta forma, el método original de Bouix et al. (2005), produce un esqueleto con 3 características: delgado, conectado, y centrado. Este tipo de esqueleto es usado para poder reconstruir el objeto original del cual se ha generado.

Bouix et al. (2005) presenta un método que a partir de una imagen binaria produce un conjunto de elementos que conforman el esqueleto en 3 dimensiones o también denominado como superficie media o *medial surface* (ver la Figura 3.4). En 3 dimensiones,

la superficie media se puede definir como el lugar de los centros de las máximas esferas inscritas, mientras que en 2 dimensiones, el eje medio o *medial axis*, corresponde al lugar de los centros de los máximos círculos inscritos.

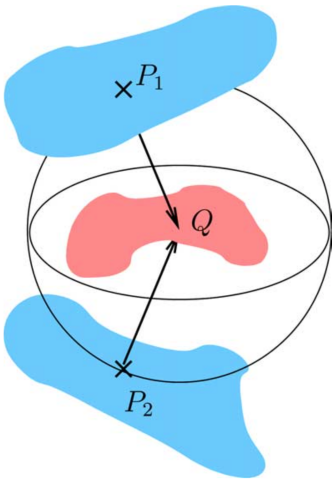


Figura 3.4: (Bouix et al., 2005). Una superficie media está resaltada en color rojo, y las 2 superficies a las que pertenece están resaltadas con el color azul. Cada punto Q en la superficie media está asociado a los 2 puntos más cercanos P_1, P_2 en la superficie del objeto. El ángulo formado con el objeto es la mitad entre los ángulos de los vectores P_1Q y P_2Q en el plano que pasa por P_1, P_2 y Q.

$$\frac{\partial S_u}{\partial t} = \hat{N} \tag{3.7}$$

Para definir este método, se considera la Ecuación de propagación de Blum (Ecuación 3.7), la cual actúa en una superficie cerrada en 3 dimensiones, simbolizada por S_u, tal que cada punto en el borde se mueve a una velocidad constante, t, hacia la dirección de la normal que apunta al interior, representada por el símbolo $\hat{N}$. Considerando el símbolo D_M, como el mapa de distancia a la superficie inicial, la cual es representada por el símbolo L_0. La magnitud de su gradiente $||\nabla D_M||$, es igual a 1. Dado $q = (x, y, z)$, $p = (D_{M_x}, D_{M_y}, D_{M_z})$ y $||p|| = 1$, el sistema Hamiltoniano está dado por la Ecuación 3.8, la cual está asociada a la función Hamiltoniana $H = 1 + ||\nabla D_M||$

$$\dot{q} = (0, 0, 0), \dot{q} = (D_{M_x}, D_{M_y}, D_{M_z}) \tag{3.8}$$

Para distinguir entre los puntos de la superficie medial de la no medial se puede realizar el cálculo del flujo promedio exterior o AOF del campo vectorial $\dot{q}$ en cada punto. Este valor puede ser obtenido por la Ecuación 3.9:

$$AOF(\dot{q}) = \frac{\int_{\delta R} < \dot{q}, \hat{N}_0 > dS_u}{area(\delta R)} \tag{3.9}$$

Donde dS_u es un elemento de la superficie δR de un volumen R, y $\hat{N}_0$ es la normal al exterior en cada punto de la superficie. Siddiqi et al. (2002) y Dimitrov et al. (2003) demuestran que mientras un volumen esférico se va reduciendo a un punto fuera de la superficie media, el flujo promedio exterior a través de su superficie, se aproxima a cero.

En cambio, cuando dicho volumen se reduce a un punto que pertenece a la superficie media, el flujo promedio exterior se aproxima a un número estrictamente negativo.

A continuación, se describen los pasos del trabajo de Bouix et al. (2005), para obtener el esqueleto a partir de una frontera inicial.

3.2.1 **Generación del mapa de distancia**. El algoritmo original propuesto por Bouix et al. (2005), utiliza el trabajo de Meijster et al. (2002) para generar un mapa de distancia a partir de una frontera en una imagen de forma paralela. Dada una imagen binaria en 2 dimensiones I_b, donde los valores pueden ser 0 o 1, usando el valor 0 para representar el fondo y el valor 1 para representar el objeto. El mapa de distancia de I_b está dado por la Ecuación:

$$D_M(x,y) = \begin{cases} 0 & \text{si } I_b(x,y) = 1 \\ min([x-x_0, y-y_0], \forall I_b(x_0,y_0) = 1) & \text{si } I_b(x,y) = 0 \end{cases} \tag{3.10}$$

Donde $[\cdot,\cdot]$ es alguna métrica de distancia. El trabajo de Meijster et al. (2002), genera D_M a partir de 2 fases o escaneos en 2 dimensiones y 3 fases en 3 dimensiones. En cada fase el algoritmo realiza un barrido desde un lado de la imagen hacia el otro. De esta forma, en la primera fase, propaga las distancias de la frontera a través de barridos desde la izquierda hacia la derecha de la imagen y en sentido contrario. De la misma forma, en la segunda fase, recorre la imagen desde abajo hacia arriba y visceversa. En 3 dimensiones, se realiza la misma operación en la dirección adicional. Como se trata de un proceso de barrido, en cada dirección se puede utilizar un número de *threads* para acelerar la ejecución del algoritmo. En este trabajo utilizamos la implementación Meijster et al. (2002) para generar el mapa de distancia, porque es un método que puede ser extendido de 2 a 3 dimensiones y puede ser implementado de forma paralela en CPU.

3.2.2 **Construcción del flujo exterior promedio**. Para encontrar el flujo exterior promedio o AOF, se necesita calcular la gradiente a partir del mapa de distancia. La gradiente del mapa de distancia (D_M) en 2 dimensiones, se puede expresar a través de la Ecuación 3.11.

$$\bigtriangledown D_M = \begin{bmatrix} G'_x \\ G'_y \end{bmatrix} = \begin{bmatrix} \frac{\partial D_M}{\partial x} \\ \frac{\partial D_M}{\partial y} \end{bmatrix} \tag{3.11}$$

Donde $\frac{\partial D_M}{\partial x}$ y $\frac{\partial D_M}{\partial y}$ son la gradiente (G') en las direcciones x y y, respectivamente. De esta forma, encontrar la gradiente de una imagen se transforma en una operación de convolución, la cual puede ser implementada de forma paralela en *CPU* y en *GPU* y además puede ser extendida de 2 a 3 dimensiones. Con la información de la gradiente, el flujo exterior promedio es construído resolviendo la Ecuación 3.9, obteniendo la Ecuación 3.12.

$$AOF(w) = \frac{1}{n} \sum_{i=1}^{n} < \hat{N}_i, \bigtriangledown D_M(w_i) > \tag{3.12}$$

Donde w_i es uno de los n vecinos de w, en 2 dimensiones $n = 8$ y en 3 dimensiones $n = 26$, y $\hat{N}_i$ es la normal exterior en w_i de la esfera unitaria con centro en w. Esta Ecuación puede ser extendida de 2 a 3 dimensiones y el procedimiento de calcular el flujo exterior promedio puede ser implementado de forma paralela, debido a que la Ecuación 3.12 es resuelta en cada elemento de la gradiente $\bigtriangledown D_M$.

3.2.3 **Adelgazamiento**. Finalmente, la idea propuesta por Bouix et al. (2005) es utilizar el mapa generado para dirigir un proceso de adelgazamiento o *thinning*. Para esto, el mapa del flujo exterior promedio obtenido es utilizado para detectar puntos que al ser eliminados, no alteran la topología del objeto. Tales puntos reciben el nombre de puntos simples. En 2 dimensiones, el proceso de adelgazamiento considera una máscara de 8 vecinos. Durante este proceso, un punto puede ser eliminado si no desconecta el objeto o genera un espacio hueco. En 3 dimensiones, la máscara consiste de 6 caras, 12 aristas y 8 vértices, y un punto es eliminado si no desconecta el objeto, produce una cavidad o espacio hueco. De esta forma, a través de un proceso iterativo se obtiene finalmente un conjunto de puntos que preservan la información de la forma del objeto.

En esta Sección se ha descrito el método de esqueletización utilizado en esta investigación, a partir del trabajo de Bouix et al. (2005), el cual consta de 3 etapas que pueden ser extendidas para trabajar con imágenes de 2 y 3 dimensiones.

A continuación, en la siguiente Sección, se va a describir el método utilizado en esta investigación para distribuir puntos usando discos de Poisson.

3.3. Muestreo de discos de Poisson

En esta Sección, se describe el método propuesto por Ying et al. (2013) para distribuir puntos sobre una nube de puntos, usando discos de Poisson. Este método a diferencia de otros como Bowers et al. (2010), no propone distribuir el espacio para procesarlo de forma paralela y evitar conflictos entre muestras. En cambio, Ying et al. (2013) utiliza una prioridad que es utilizada para resolver conflictos. En general, el elemento con mayor prioridad es aceptado como parte de la muestra final. La ventaja de este método, para nuestro propósito, es que puede ser implementado de forma paralela con *threads* de CPU y GPU.

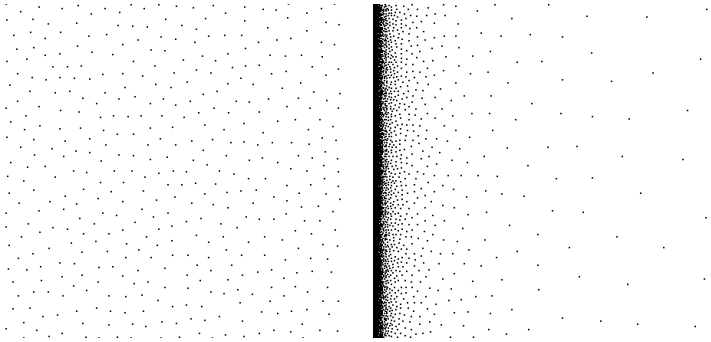
Dado un dominio de muestreo D_P, las muestras obtenidas a través de discos de Poisson son un conjunto $X = X_i \in D; i = 1, 2, ..., N_P$ compuesto de N_P muestras, las cuales deben cumplir con las propiedades de la Ecuación 3.13 y la Ecuación 3.14.

$$\forall x_i \in X, \forall M \subseteq D_P : Prob(x_i \in M) = \int_M dx \tag{3.13}$$

$$\forall x_i, x_j \in X : ||x_i - x_j|| \geq 2r \tag{3.14}$$

Donde $Prob(\cdot)$ es una probabilidad condicional. La primera condición (Ecuación 3.13) establece que una muestra aleatoria distribuida de manera uniforme, x_i que forma parte de X, tiene una probabilidad de estar en el interior de un conjunto M, que forma parte de D_P, igual al hiper-volumen de M. La segunda condición establece que 2 muestras deben estar separadas por una distancia de $2r$.

La Figura 3.5, muestra un ejemplo de una distribución de puntos de forma uniforme y adaptativa utilizando discos de Poisson. La Figura 3.5a distribuye puntos utilizando un radio constante. En cambio, la Figura 3.5b distribuye puntos utilizando una función.



(a) Muestreo de discos de Poisson uniforme utilizando un radio constante.

(b) Muestreo de discos de Poisson adaptativo.

Figura 3.5: Muestras uniformes (a) y adaptativas (b) generadas a través de discos de Poisson usando el trabajo de (Ying et al., 2013).

El método propuesto por Ying et al. (2013), está inspirado en el método clásico *Dart Throwing* (Cook, 1986), donde se observa que cada muestra tiene una posición aleatoria, así como un momento de nacimiento. Mientras las iteraciones transcurren, las muestras son aceptadas o rechazadas, de acuerdo a su distancia a las muestras que ya han sido previamente aceptadas. En base a esta observación, Ying et al. (2013) asigna un valor numérico único a cada elemento de una nube de puntos, la cual representa la prioridad de la muestra. Este algoritmo puede procesar varias muestras al mismo tiempo, dependiendo de la cantidad de *threads* disponibles. Cuando existe algún conflicto entre las muestras que están siendo procesadas, las que tengan mayor prioridad son aceptadas, como se observa en la Figura 3.6. Esta prioridad se define en la Ecuación 3.15. Donde k es el número de *threads* y p es el conjunto de muestras.

$$prioridad(p_i) = \frac{rand() * k + i}{RAND\_MAX * T} \tag{3.15}$$

Para la implementación del método, Ying et al. (2013) propone que una muestra tenga un estado, el cual puede ser: "pendiente", significa que la muestra no ha sido procesada por ningún *thread*; "activo", indica que una muestra está siendo procesada por un *thread*; "aceptado", incluye la muestra como parte del resultado o "rechazado", evita que esa muestra sea considerada en las siguientes iteraciones por algún *thread*, reduciendo el número de elementos a procesar.

En un inicio, cada elemento de la nube de puntos es inicializado con el estado "pendiente" y durante cada iteración, el algoritmo selecciona k muestras de manera aleatoria,

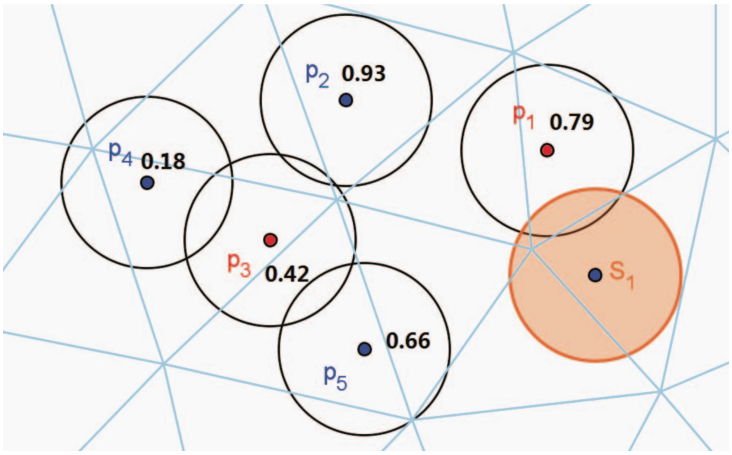


Figura 3.6: (Ying et al., 2013). Ilustración de la forma en que se resuelve un conflicto. s_1 es una muestra aceptada y hay 5 *threads* disponibles. Cada *thread* procesa una muestra activa, p_i, que tienen una prioridad aleatoria y única. El algoritmo acepta p_2, p_4, p_5 y rechaza p_1, p_3

las cuales pasan a un estado "activo". Cada *thread*, procesa un punto, p_i, y busca los puntos "activos" y "pendientes" dentro de un radio, $radius(p_i, 2r)$. Dentro de este radio, se controlan los conflictos con otros puntos que tengan el estado "activo", usando la prioridad asignada a cada uno. Si p_i es aceptado, entonces, todos los puntos que se encuentran dentro del radio $radius(p_i, 2r)$, que tengan estado "pendiente" cambian al estado "rechazado". Este procedimiento se repite hasta que todos los elementos con el estado "pendiente" sean procesados.

Como cualquier otro método basado en discos de Poisson, Ying et al. (2013) debe cumplir con las 2 condiciones expuestas en la Ecuación 3.13 y en la Figura 3.14. El trabajo de Ying et al. (2013), satisface la segunda condición (Ecuación 3.14) ya que usa la distancia geodésica (Yang y Cohen, 2016) al tratar con superficies y euclideana al procesar una nube de puntos.

A continuación, se demuestra que el algoritmo propuesto por Ying et al. (2013) cumple con la primera condición (Ecuación 3.13) indicada anteriormente. Siendo $S = \{s_1, ..., s_\omega\}$, un conjunto que contiene ω muestras de discos de Poisson generados por el método *Dart Throwing*. Siendo a su vez, $p_i, i = 1, .., k$ los puntos que son aleatoriamente distribuidos de manera uniforme sobre D_P. Si se desordenan los índices $\{1, ..., k\}$, y luego se añaden los k puntos desordenados de manera secuencial a S, comprobando la restricción de la mínima distancia, el resultado es un conjunto de discos de Poisson, ya que el procedimiento es el mismo que *Dart Throwing*. Cada punto tiene un orden único, el cual está dado por el índice del punto.

El método de Ying et al. (2013) construye una nube densa de puntos P a partir de una superficie, tal que los puntos sean aleatorios y distribuidos de manera uniforme en D_P. Donde cada punto p_i tiene un valor de prioridad único y aleatorio, el cual es equivalente a desordenar los índices $\{1, ...|P|\}$. Por lo tanto, se pueden procesar k puntos simultáneamente, y el resultado contiene una distribución de discos de Poisson uniforme y aleatoria. Hay que precisar que primero se desordena los índices $\{1, ...|P|\}$ y luego se dividen en k grupos del mismo tamaño. Durante la ejecución del algoritmo, cada *thread* escoge un índice de cada grupo. Así, se debe probar que esta selección es imparcial o *unbiased*. Para probar esto, se define P_i^p como el número de muestras en estado "pendiente" disponibles en el grupo i, y $I = \sum_{i=1}^{k} P_i^p$ el número total de muestras en estado "pendiente"

disponibles en todos los grupos. Sin dividir los grupos, la probabilidad de seleccionar una muestra "pendiente" es $1/I$. Dividiendo los grupos, la probabilidad de una muestra arbitraria q del grupo i es P_i^p/I. También, la probabilidad de q de ser seleccionada por el thread i es $1/P_i^p$. Por lo tanto, la probabilidad de seleccionar q es igual a $1/I$, lo que implica que la selección basada en particiones es imparcial.

En esta Sección se ha descrito el método de Ying et al. (2013), el cual utiliza un valor numérico que indica prioridad entre muestras, para resolver conflictos de forma paralela. Además se ha demostrado que este método cumple con las propiedades de un conjunto de muestras obtenidas a través de discos de Poisson.

Durante este Capítulo, se ha descrito los métodos que van a ser utilizados para definir la propuesta de nuestra investigación, en el siguiente Capítulo. El método propuesto por Cuadros-Vargas et al. (2009) es analizado para basar nuestra propuesta en su método de construcción de mallas a partir de imágenes. El trabajo de Bouix et al. (2005) es utilizado para extraer el esqueleto de una frontera en una imagen y el algoritmo de Ying et al. (2013), es usado para distribuir un conjunto de puntos a través de discos de Poisson en 2 y 3 dimensiones de forma paralela.

Capítulo 4

Algoritmo $kMesh$

En este Capítulo, describimos la propuesta de esta investigación basada en los métodos descritos anteriormente en el Capítulo 3. Nuestra propuesta, denominada $kMesh$, está dividida en 3 etapas, como se puede observar en la Figura 4.1:

(I) **Mapeo de densidad**, descrita en la Sección 4.1, recibe una imagen en 2 o 3 dimensiones ($\mathcal{IMG}$) que, de ser necesario, se convierte a escala de grises ($\mathcal{IMG}_{gray}$). Con la imagen en escala de grises, se obtiene un borde para definir una frontera ($\mathcal{B}$) y se extrae su esqueleto ($\mathcal{E}$) en base al trabajo de Bouix et al. (2005). Después, el borde y el esqueleto son usados para calcular un valor numérico, al que denominaremos densidad, para cada punto del borde. Esta densidad en el borde ($\mathcal{B}'$) proporciona información acerca de la forma y proximidad de los elementos que conforman el objeto. Esta primera etapa culmina cuando la densidad en el borde, se expande hacia el interior de la imagen, en una representación que denominaremos como Mapa de densidad ($\mathcal{D}$).

(II) **Muestreo**, detallada en la Sección 4.2, utiliza la densidad en el borde $\mathcal{B}'$ y la densidad en el interior $\mathcal{D}$, generadas en la etapa de Mapeo de densidad, para insertar puntos sobre el borde ($\mathcal{S}_\mathcal{B}$) y el interior ($\mathcal{S}_\mathcal{I}$) de la imagen respectivamente. Para distribuir estos puntos sobre la imagen utilizamos el trabajo propuesto por Ying et al. (2013). De esta forma, se espera conseguir una nube de puntos adaptable a la forma del objeto.

(III) **Modelamiento**, explicada en la Sección 4.3, genera una representación geométrica usando la nube de puntos de la etapa de Muestreo ($\mathcal{S}_\mathcal{B}$ y $\mathcal{S}_\mathcal{I}$). Esta etapa construye 2 mallas, la primera, denominada $kMesh_{Cm}$, utiliza los puntos insertados sobre el borde, para cumplir con el segundo objetivo de nuestra propuesta, mencionado en el Capítulo 1; y la segunda, denominada $kMesh_{Im}$, utiliza los puntos insertados en el interior de la imagen, para cumplir con el tercer objetivo de nuestra propuesta. Esta última, incluye criterios de calidad de malla basados en la forma del objeto.

A continuación, describimos en detalle estas tres etapas.

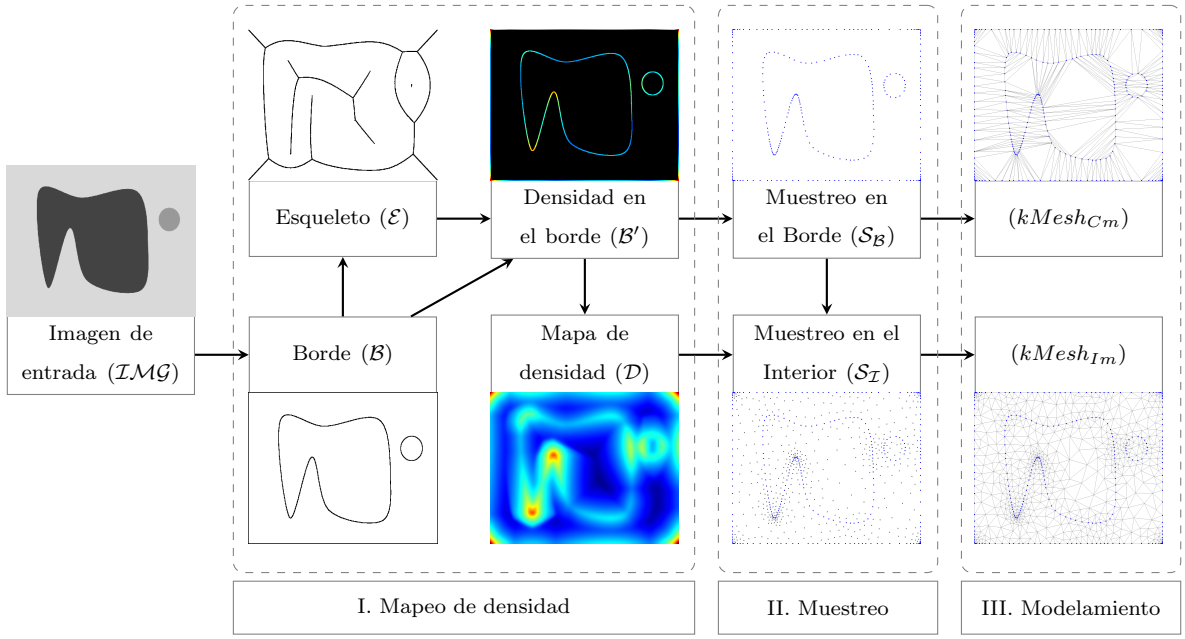


Figura 4.1: Etapas de nuestra propuesta, $kMesh$, para construir las mallas $kMesh_{Cm}$ y $kMesh_{Im}$ a partir de una imagen: (I) Mapeo de densidad, (II) Muestreo y (III) Modelamiento.

4.1. Mapeo de densidad

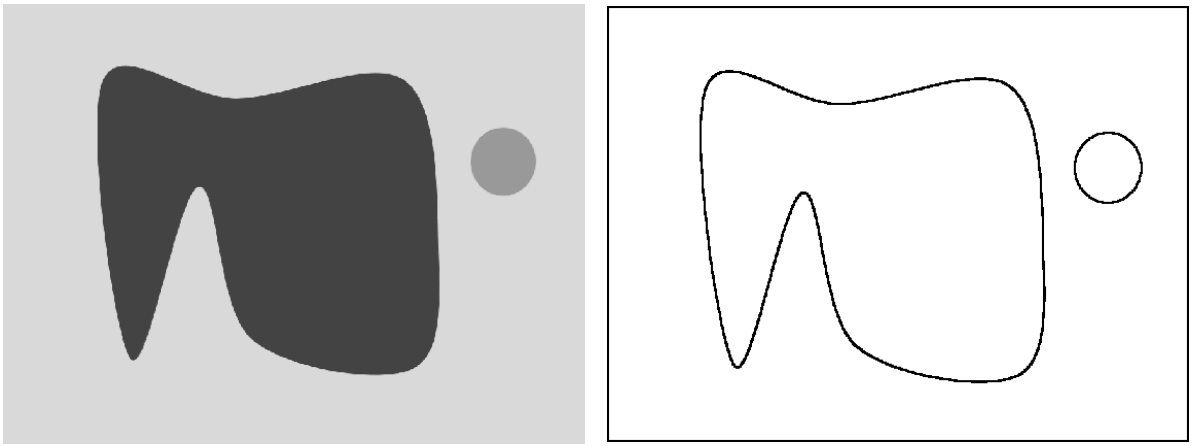
La etapa de Mapeo de densidad busca, a partir de una imagen en 2 o 3 dimensiones, generar una representación de la imagen de entrada de acuerdo a la forma del objeto, para poder identificar las zonas que tienen mayor o menor curvatura. A continuación, describiremos las partes de esta etapa de Mapeo de Densidad.

4.1.1. Selección de borde ($\mathcal{B}$)

El objetivo de esta primera parte es definir una frontera sobre la imagen. Para lograr esto, nuestro trabajo, al igual que el método $Imesh$, utiliza un conjunto de isovalores sobre una imagen en escala de grises, para determinar el borde o frontera del objeto. Estos isovalores están comprendidos entre los valores enteros 0 y 255.

De esta forma, lo primero que efectuamos, es convertir la imagen de entrada ($\mathcal{IMG}$) a una representación en escala de grises, simbolizada por $\mathcal{IMG}_{gray}$ (ver Figura 4.2a), y resolver para cada elemento de $\mathcal{IMG}_{gray}$ la Ecuación 3.4, como se ha descrito en la Sección 3.1. Este cálculo se realiza para cada elemento de $\mathcal{IMG}_{gray}$, consiguiendo de esta forma, una complejidad algorítmica de $\mathcal{O}(\eta)$, en modo secuencial, donde η es el número total de elementos de la imagen en 2 o 3 dimensiones. Sin embargo, como se trata de un proceso independiente para cada elemento, este método puede ser implementado de forma paralela (*CPU*, *GPU*) utilizando k *threads*.

A través del procedimiento descrito, se obtiene el borde o frontera del objeto, representado por el símbolo $\mathcal{B}$, y el interior representado por el símbolo $\mathcal{I}$. El interior corresponde a las partes de la imagen que no pertenecen al borde. Adicionalmente, estamos incluyendo el marco de la imagen como parte del borde como se puede observar en la Figura 4.2b. En este ejemplo, se está utilizando los isovalores 150 y 200 para definir el borde $\mathcal{B}_{\{150,200\}}$, incluyendo el marco de la imagen, presentado en color negro y el interior $\mathcal{I}$, presentado en color blanco.



(a) $\mathcal{IMG}_{gray}$. Imagen en 2 dimensiones de tamaño $877x677$.

(b) Borde ($\mathcal{B}_{\{150,200\}}$), en color negro e interior ($\mathcal{I}$), en color blanco, definidos con los isovalores 150 y 200.

Figura 4.2: Selección del borde ($\mathcal{B}$) y el interior ($\mathcal{I}$) de la imagen en escala de grises (a), usando los isovalores 150 y 200. Adicionalmente, el marco de la imagen también es considerado como parte del borde.

El borde obtenido por un conjunto de isovalores, obtiene un objeto cerrado donde el grosor de la frontera es delgado. Como se trata de un proceso de umbralización, distintos isovalores pueden seleccionar diferentes partes del objeto. Entonces, para encontrar un conjunto de isovalores apropiado, se debe elaborar una serie de pruebas con distintos isovalores hasta encontrar el objeto con el cual se quiere trabajar. De esta manera, el primer parámetro de nuestra propuesta, es el conjunto de isovalores que definen la frontera del objeto. El análisis de este parámetro se explica más adelante, en la Sección 5.4.1.

Por otro lado, se pueden utilizar otros métodos de selección de borde como Canny o Prewitt para definir la frontera del objeto como se indica en la Sección 3.1. Sin embargo, el método $Imesh$, obtiene mejores resultados cuando selecciona la frontera con un conjunto de isovalores, porque son usados para realizar una mejor partición de la malla producida en $Imesh_{Cm}$. Ya que nuestra propuesta se ha basado en este método, conservaremos este algoritmo para definir la frontera del objeto. Pero, nuestra propuesta puede usar cualquier otro método de selección de borde.

A partir del borde o frontera, el siguiente paso, explicado en la Sección 4.1.2, consiste en extraer información de la forma del objeto, a través de su esqueleto.

4.1.2. Extracción del esqueleto ($\mathcal{E}$)

Uno de los objetivos de este trabajo es construir una malla que se adapte mejor a los detalles del objeto. A diferencia del método *Imesh*, nuestra propuesta busca utilizar información de forma como parte de nuestro proceso de construcción de malla. Para lograr este propósito, utilizamos un descriptor basado en el esqueleto del objeto.

Como se ha mencionado en la Sección 2.3, existen varios métodos de esqueletización, con diferentes características. Nuestra propuesta está basada en el método propuesto por Bouix et al. (2005) principalmente porque es un método altamente paralelizable. El método de Bouix et al. (2005) sigue las siguientes etapas: generación del mapa de distancias, cálculo del flujo de la gradiente y adelgazamiento. Sin embargo, nuestra propuesta reemplaza la última etapa por un proceso de umbralización. De esta forma, nuestra propuesta para obtener el esqueleto, está dividida en 3 partes: elaboración del mapa de distancia, cálculo del flujo de la gradiente y umbralización, los cuales se describen a continuación.

4.1.2.1. Elaboración del mapa de distancias

En esta parte, a partir de una región de interés, se calcula un mapa de distancia. Esta región de interés está dada por el borde $\mathcal{B}$, obtenido anteriormente por la etapa de selección de borde en la Sección 4.1.1. A partir de este borde, se elabora una propagación por barridos de acuerdo al método propuesto por Meijster et al. (2002), como se describe en la Sección 3.2.1. Este método ofrece una implementación lineal de orden $\mathcal{O}(\eta)$, donde η es el número total de elementos de la imagen. Usando este método, cada dirección de la imagen puede ser dividida en k filas que pueden ser procesadas de forma paralela. Además, en 3 dimensiones, se realiza el mismo procedimiento considerando la dirección adicional de la imagen. El resultado de esta propagación se puede observar en la Figura 4.3.

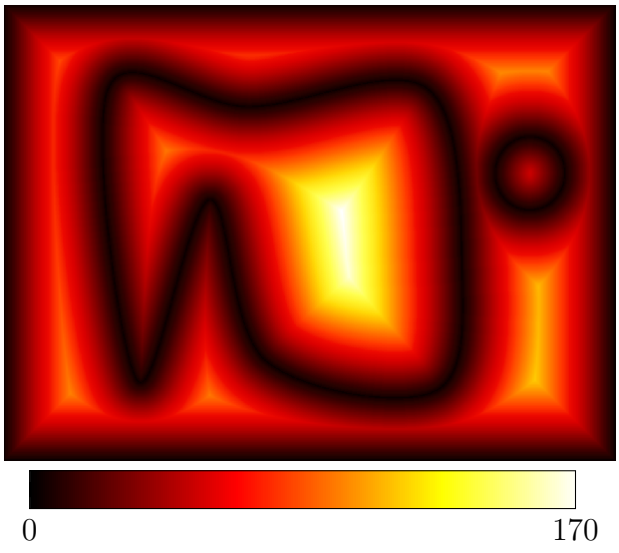


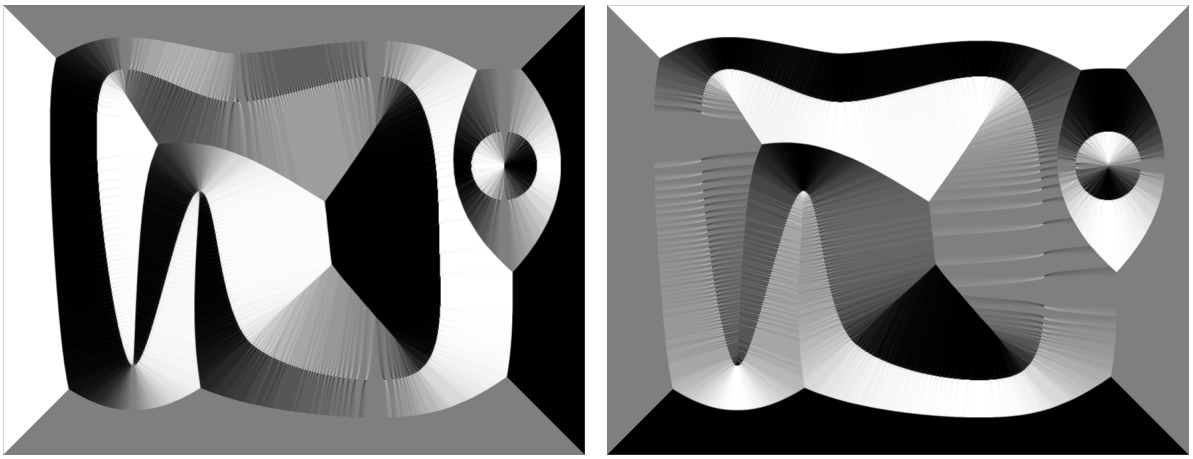
Figura 4.3: Mapa de distancia generado a partir del borde $\mathcal{B}$, obtenido anteriormente en la Sección 4.1.1. El mínimo valor, correspondiente a los elementos de $\mathcal{B}$, es 0 y el máximo valor, correspondiente al elemento más alejado de $\mathcal{B}$, es 170. El esqueleto $\mathcal{E}$ está definido en las discontinuidades del mapa de distancia.

En una analogía, podemos considerar la imagen como un medio de propagación, y cada elemento del borde como una pequeña mecha de fuego, la cual se expande a una

velocidad constante por la imagen. El esqueleto, entonces, se ubica en las zonas donde los frentes de fuego se encuentran. Como se puede observar en la Figura 4.3, el mapa de distancia es una función que asocia cada posición de la imagen con un valor numérico, el cual representa su mínima distancia hacia el borde. Desde esta perspectiva, el esqueleto se ubica en las crestas o *ridges* de la función. Por esta razón, los siguientes pasos, buscan ubicar los elementos que conforman el esqueleto.

4.1.2.2. Cálculo del flujo de la gradiente

A partir del mapa de distancia obtenido, se busca localizar las crestas que definen el esqueleto de la imagen. Encontrar el esqueleto, entonces, se convierte en un problema de encontrar las singularidades en el mapa de distancia generado. Para esto, utilizamos la gradiente del mapa de distancia, como se ha indicado en la Sección 3.2.2, para detectar los cambios en el incremento de los valores del mapa de distancia con respecto a cada una de las direcciones de la imagen.



(a) Gradiente del mapa de distancia con respecto a la dirección x.

(b) Gradiente del mapa de distancia con respecto a la dirección y.

Figura 4.4: Componentes de la gradiente del mapa de distancia obtenido en la Sección 4.1.2.1.

La gradiente es una representación que señala el cambio de los valores de una función escalar definida, en este caso, por el mapa de distancia. La gradiente puede ser calculada a través de máscaras, que detectan el cambio de los valores, para cada dimensión de la imagen. De acuerdo a esto, la operación de obtener la gradiente a partir del mapa de distancia, tiene una complejidad algorítmica de orden $\mathcal{O}(\eta)$, donde η, es el número total de elementos que conforman la imagen. Ya que el proceso de convolución puede realizarse de forma independiente para cada elemento, se puede utilizar k *threads* para implementar esta parte de forma paralela. Las gradientes obtenidas en esta parte, se pueden observar en la Figura 4.4.

Después, de acuerdo a la Sección 3.2.2, se calcula el flujo exterior promedio o AOF, como se puede ver en la Figura 4.5, a partir de la gradiente. Este proceso produce una

representación escalar de valores con signo, que denotan la cantidad de "flujo" entrando o saliendo de un elemento. Este valor es calculado para cada elemento de la gradiente, según la Ecuación 3.12. Así, calcular el flujo exterior promedio, tiene una complejidad algorítmica de $\mathcal{O}(\eta)$, donde η es el número total de elementos de la imagen. A su vez, este es un procedimiento que puede realizarse de forma independiente, por lo que se puede paralelizar dividiendo la imagen en k partes, que pueden ser procesadas de forma paralela. La Figura 4.5, muestra el resultado de calcular el flujo exterior promedio a partir de la gradiente.

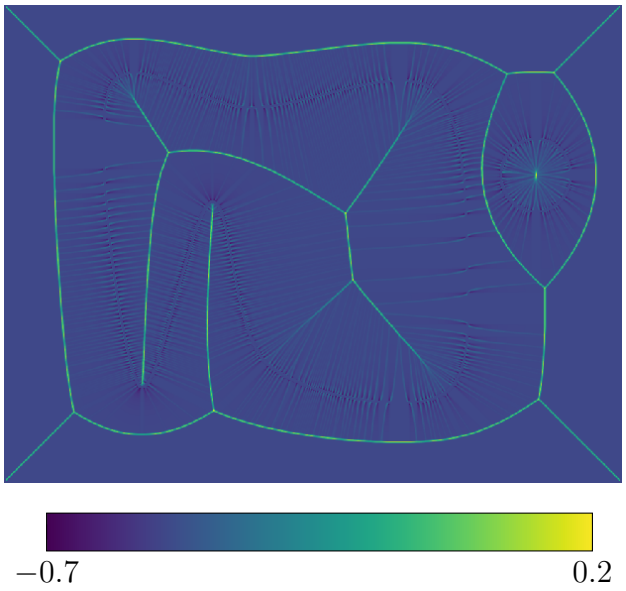


Figura 4.5: Representación del flujo exterior promedio o AOF, en base a la gradiente. Cada valor es calculado a partir de la Ecuación 3.12, como se indica en el trabajo de Bouix et al. (2005).

4.1.2.3. Umbralización del flujo de la gradiente

Sobre el flujo exterior promedio o AOF, esta última parte del proceso de esqueletización, utiliza un umbral para separar los elementos que conforman el esqueleto. Como se ha mencionado, el esqueleto se encuentra a través de un valor negativo sobre el flujo exterior promedio obtenido. En este ejemplo, estamos utilizando el umbral -0.4, para extraer el esqueleto que se observa en la Figura 4.6.

Este proceso de umbralización, es una consulta hecha para cada elemento, que conforma el flujo exterior promedio y puede realizarse de forma independiente. Por esta razón, tiene una complejidad algorítmica de $\mathcal{O}(\eta)$, donde η, es el número total de elementos del flujo exterior promedio. De la misma manera, puede ser paralelizado dividiendo la imagen en k partes, usando la cantidad de *threads* disponibles.

El umbral utilizado en esta parte, es el segundo parámetro requerido por nuestra propuesta. Diferentes valores, extraen distintos esqueletos. El análisis de este parámetro se realiza más adelante, en la Sección 5.4.2. A continuación, en la Sección 4.1.3, se utilizará la frontera del objeto y el esqueleto extraído, para tener información acerca de la forma del objeto.

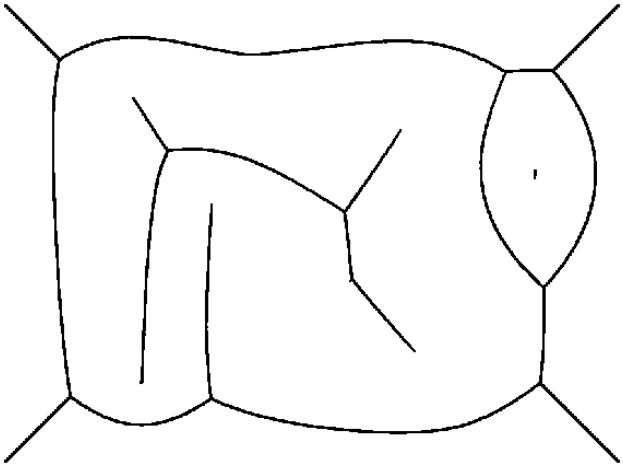


Figura 4.6: $\mathcal{E}_{-0,4}$. Esqueleto obtenido aplicando un umbral negativo, con el valor -0.4 sobre el flujo exterior promedio de la gradiente de la Figura 4.5.

4.1.3. Densidad en el borde ($\mathcal{B}'$)

El segundo objetivo de este trabajo es incluir la forma del objeto en el proceso de construcción de la malla. La manera en cómo se propone agregar esta información es a través de un valor numérico, denominado densidad, para cada punto del borde $\mathcal{B}$. Esta representación, en el borde, es simbolizada por $\mathcal{B}'$. La densidad, proporciona información de 2 aspectos del borde: curvatura y proximidad, los cuales son utilizados para reconocer las zonas del borde, que deben ser representadas por la malla con mayor o menor detalle, insertando más o menos cantidad de elementos, respectivamente. Cuando algunas partes del borde se encuentran muy cerca, se necesita que la malla incluya más elementos para poder diferenciar entre esas zonas del objeto. Por otro lado, cuando algunas partes de borde son rectas, es decir, presentan poca curvatura, el número de elementos que la malla necesita en esas zonas, es menor.

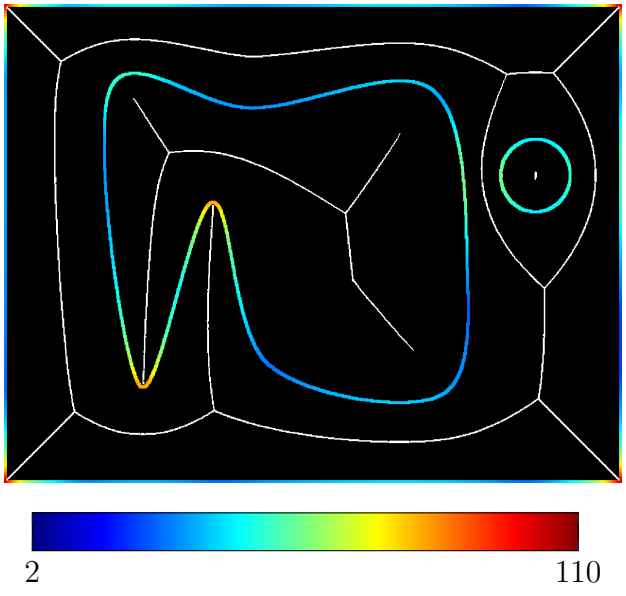


Figura 4.7: Densidad en el borde ($\mathcal{B}_2'^{110}$), a partir del esqueleto extraído $\mathcal{E}_{-0,4}$. Los colores que tienden a rojo indican zonas con mucha curvatura, cercanas a otras partes del borde. Los colores que tienden al azul indican zonas con menos curvatura, alejadas de otras partes del borde.

De esta forma, en la Figura 4.7 se puede visualizar a través de un mapa de colores, los diferentes niveles de curvatura y proximidad entre los elementos del borde. Los colores

que tienden a rojo indican partes del borde con mucha curvatura, cercanas a otras partes del borde, y los colores que tienden al azul indican partes del borde con poca curvatura, alejadas de otras partes del borde.

Para calcular la densidad, utilizamos el borde y el esqueleto extraído en las etapas anteriores. El valor numérico en cada elemento del borde, corresponde a la distancia al punto más cercano del esqueleto. Para obtener estos valores, se ha utilizado una estructura $kdTree$, que en primer lugar, es construida insertando todos los puntos que conforman el esqueleto $\mathcal{E}_{-0,4}$. Construir esta estructura $kdTree$ tiene una complejidad de $\mathcal{O}(\eta log(\eta))$, donde η, es la cantidad de elementos que pertenecen al esqueleto. Luego, de forma paralela usando k *threads*, se consulta para cada punto del borde $\mathcal{B}$, su distancia al punto más cercano del esqueleto, obteniendo así una representación denominada Densidad en el borde ($\mathcal{B}'$).

Con la Densidad en el borde ($\mathcal{B}'$), se ha conseguido expresar los criterios de curvatura y proximidad de la frontera del objeto, a través de un valor numérico, usando el esqueleto extraído en la Sección 4.1.2. De esta forma, se puede identificar las partes del borde que necesitan ser representadas por la malla con mayor o menor detalle.

Los valores de la Densidad en el borde, como se describe a continuación, en la Sección 4.1.4, son expandidos hacia el resto de la imagen, para generar una malla con elementos no sólo en el borde ($\mathcal{B}$) sino también en el interior ($\mathcal{I}$) de la imagen, incluyendo criterios de calidad, como se explicará más adelante.

4.1.4. Mapa de densidad ($\mathcal{D}$)

En este punto de nuestra investigación, se busca representar toda la imagen considerando la información obtenida en el borde. Para cumplir con este propósito, nuestra propuesta consiste en expandir la Densidad en el borde $\mathcal{B}'$ de forma incremental hacia toda la imagen, a través de un recorrido por amplitud, similar a una propagación por frentes de onda, considerando el borde como frente de propagación inicial.

Este proceso de propagación genera una representación de la imagen que denominaremos como Mapa de densidad y está simbolizado por $\mathcal{D}$. Generar este recorrido en amplitud demora $\mathcal{O}(\eta)$, donde η es el número de elementos de la imagen. Cada iteración, está compuesta por un frente de propagación, el cual puede ser dividido en k partes y puede ser procesado de forma paralela. El resultado de la propagación de los valores a partir de la Densidad en el borde $\mathcal{B}'$, se puede observar en la Figura 4.8.

En esta primera etapa de nuestra propuesta, denominada Mapeo de densidad, como se ha explicado, se busca representar la imagen a través de un valor numérico, denominado densidad. El valor de esta densidad, gracias al esqueleto ($\mathcal{E}_{-0,4}$), permite considerar la forma del objeto seleccionado ($\mathcal{B}$). De esta forma, se puede identificar las zonas de la imagen donde la malla debe contener mayor detalle. Esto sucede por ejemplo en las regiones del borde que tienen mayor curvatura y se encuentran muy cerca; y por otro lado, se puede detectar las zonas de la imagen que deben ser representadas por la malla con

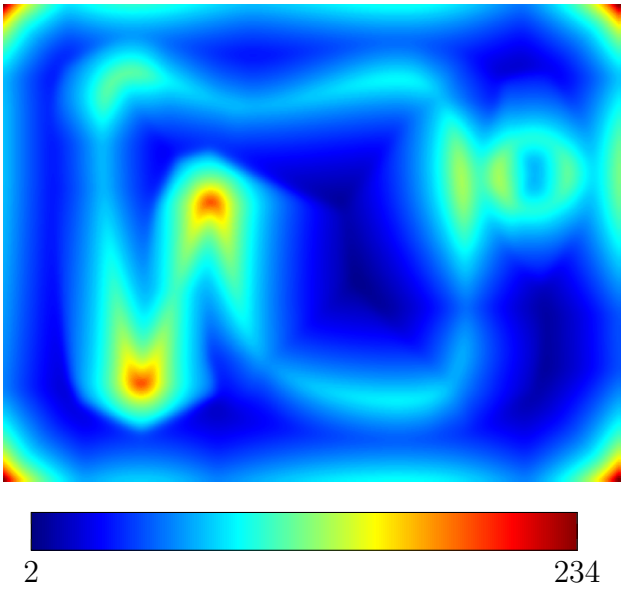


Figura 4.8: Mapa de densidad ($\mathcal{D}$) generado a partir de la Densidad en el borde. El mínimo valor es 2 y el máximo valor es 234, representado por el símbolo $\mathcal{D}_2^{234}$

menor detalle, lo que ocurre en las regiones del borde que presentan menor curvatura y se encuentran alejadas.

En cuanto a los métodos utilizados en esta etapa de Mapeo de densidad, la mayoría de métodos, procesan toda la imagen para obtener diferentes representaciones. Debido a esto, la complejidad algorítmica en esta etapa, es $\mathcal{O}(\eta)$, donde η, es el número de elementos que conforman la imagen. Es importante resaltar, que la complejidad algorítmica, en esta etapa, no se incrementa cuando procesa imágenes en 3 dimensiones, la complejidad es la misma y todos los métodos son paralelizables.

La siguiente etapa, en la Sección 4.2, utilizará esta información para insertar puntos de forma adaptativa sobre la imagen, usando el Mapa de densidad ($\mathcal{D}$) obtenido.

4.2. Muestreo

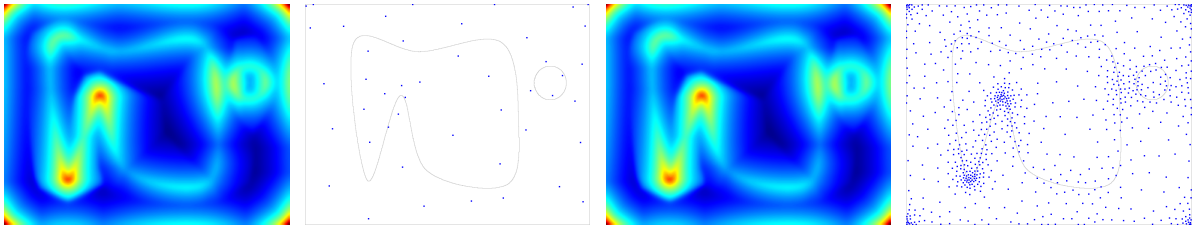
Esta segunda etapa de nuestra propuesta, se denomina Muestreo y está representado por el símbolo $\mathcal{S}$. El objetivo es distribuir un conjunto de puntos sobre la imagen, considerando el Mapa de densidad ($\mathcal{D}$) producido en la etapa de Mapeo de densidad.

Como se ha descrito en la Sección 3.3, nuestra propuesta utiliza el trabajo de Ying et al. (2013) para distribuir puntos sobre la imagen. Realizamos esto, con el objetivo de usar un conjunto de puntos como generadores de alguna representación geométrica, como por ejemplo, una triangulación. El método propuesto por Ying et al. (2013) nos permite generar una nube de puntos adaptativa, en un proceso que tiene una complejidad algorítmica de $\mathcal{O}(\eta)$, donde η, es el número de puntos generados. Además, nos permite usar k *threads* para poder implementar el algoritmo de forma paralela.

Para distribuir puntos de forma adaptativa, se necesita una función que asocie cada

posición de la imagen, con un valor numérico que indique distancia. Esta función, está basada en los Mapas de densidad ($\mathcal{B}'$ y $\mathcal{D}$) obtenidos en la primera etapa de nuestra propuesta. Sin embargo, si distribuimos puntos sobre el Mapa de densidad obtenido, conseguiremos un resultado como el que se observa en la Figura 4.9b. Los puntos distribuidos en esta imagen, utilizan el Mapa de densidad $\mathcal{D}_2^{234}$, donde el mínimo valor es 2 y el máximo valor es 234. Expresado de otra forma, usando el Mapa de densidad $\mathcal{D}_2^{234}$, indicamos que la mínima distancia entre puntos es 2 y la máxima distancia entre puntos es 234. El resultado de distribuir puntos con esta función, no es incorrecto, pero para nuestro propósito debe ser modificado.

Para obtener una mejor distribución de puntos, se realiza un proceso de normalización del Mapa de densidad $\mathcal{D}_2^{234}$, a un nuevo rango de valores. Por ejemplo, una mejor distribución de puntos, se observa en la Figura 4.9d, la cual utiliza el Mapa de densidad normalizado, donde el nuevo mínimo valor es 3 y el nuevo máximo valor es 60 y está representado por el símbolo $\mathcal{D}_3^{60}$.



(a) $\mathcal{D}_2^{234}$. Mapa de densidad obtenido en la etapa de Mapeo de densidad.

(b) Distribución de puntos con discos de Poisson usando (a).

(c) $\mathcal{D}_3^{60}$. Mapa de densidad normalizado al rango 3 y 60.

(d) Distribución de puntos con discos de Poisson usando (b).

Figura 4.9: Distribución de puntos usando discos de Poisson utilizando 2 funciones de distancia. En la primera (b), se utiliza la función obtenida por el Mapa de densidad (a). En la segunda (d), se utiliza la función normalizada al rango de 3 y 60 (c).

Realizar el proceso de normalización de acuerdo a un nuevo rango, es un procedimiento independiente que se calcula en cada elemento del Mapa de densidad, por este motivo, su complejidad algorítmica es $\mathcal{O}(\eta)$, donde η es el número de elementos del Mapa de densidad, y puede aprovechar k *threads* disponibles, para ser implementado de manera paralela. De esta forma, el nuevo rango de normalización, es el tercer y último parámetro requerido por nuestra propuesta. Más detalles sobre este parámetro, se discuten en la Sección 5.4.3.

Como se puede observar en la Figura 4.9d, la distribución de puntos con el Mapa de densidad normalizado al rango de 3 y 60, muestra una mejor distribución de puntos. Sin embargo, se puede ver que hay una poca cantidad de puntos que se encuentran sobre el borde de la imagen. Para mejorar este resultado, esta etapa se divide en 2 partes, explicadas a continuación.

4.2.1. Muestreo en el borde ($\mathcal{S}_\mathcal{B}$)

A partir del Mapa de densidad normalizado a los valores 3 y 60, simbolizado por $\mathcal{D}_3^{60}$, utilizamos el método de Ying et al. (2013), para distribuir puntos sobre el borde de la imagen, representados por el símbolo $\mathcal{S}_\mathcal{B}$. El resultado de esta distribución de puntos se puede observar en la Figura 4.10.

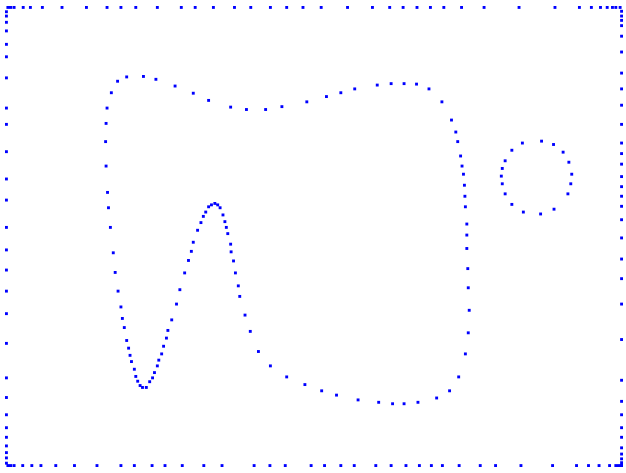


Figura 4.10: $\mathcal{S}_\mathcal{B}$. Distribución de puntos usando discos de Poisson sobre el borde de la imagen ($\mathcal{B}$) utilizando $\mathcal{D}_3^{60}$.

De esta manera aseguramos que haya una correcta distribución de puntos sobre el borde, considerando los detalles de curvatura del objeto. A partir de esta distribución de puntos en el borde, distribuimos puntos sobre el interior de la imagen, como se explica a continuación.

4.2.2. Muestreo en el interior ($\mathcal{S}_\mathcal{I}$)

A partir de los puntos distribuidos sobre el borde de la imagen $\mathcal{S}_\mathcal{B}$, se realiza el mismo procedimiento utilizando el Mapa de densidad normalizado $\mathcal{D}_3^{60}$, para generar puntos sobre el interior de la imagen, representados por el símbolo $\mathcal{S}_\mathcal{I}$. De esta forma, se puede producir una nube de puntos en toda la imagen, la cual considera la forma del objeto, para generar una mayor densidad de puntos alrededor de las zonas que presentan mayor curvatura. De igual manera, esta nube de puntos, presenta una menor densidad, alrededor de las zonas con poca curvatura, como se puede observar en la Figura 4.11.

En esta segunda etapa de la propuesta, denominada Muestreo, se ha utilizado el Mapa de densidad de la etapa de Mapeo de densidad, para identificar las zonas de la imagen que presentan mayor o menor curvatura y se encuentran más o menos alejadas de otras partes del objeto. Con esta información, hemos distribuido puntos de forma adaptativa usando el trabajo de Ying et al. (2013), para generar una nube de puntos adaptada a la forma del objeto. Este proceso de distribución de puntos, tiene la misma complejidad algorítmica en 2 y 3 dimensiones, $\mathcal{O}(\eta)$, donde η, es el número de puntos generados. Además, este algoritmo puede ser implementado de forma paralela.

Los puntos generados en esta etapa, serán usados a continuación, en la Sección 4.3, en la etapa de Modelamiento, para generar una representación geométrica.

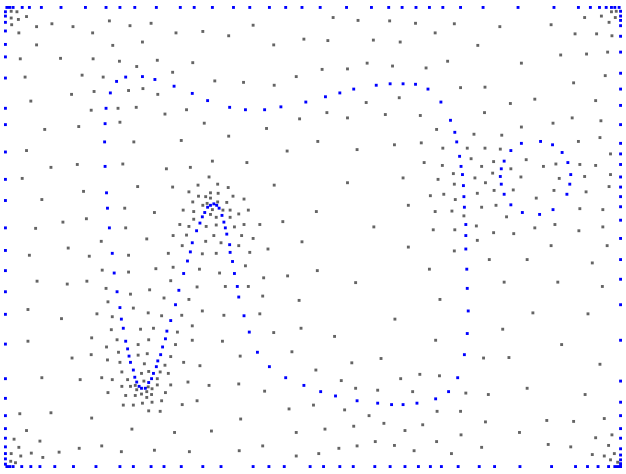


Figura 4.11: $\mathcal{S}_{\mathcal{I}}$. Distribución de puntos usando discos de Poisson sobre el interior de la imagen ($\mathcal{I}$) con la función $\mathcal{D}_3^{60}$, considerando los puntos distribuidos en el borde ($\mathcal{S}_{\mathcal{B}}$). Los puntos de color negro pertenecen al borde y los puntos de color gris al interior.

4.3. Modelamiento

Finalmente, en esta última etapa, denominada Modelamiento, nuestro objetivo es utilizar los puntos generados en la etapa de Muestreo, para generar una representación geométrica.

Debido a que nuestro método está basado en el trabajo de Cuadros-Vargas et al. (2009), en esta etapa utilizaremos la triangulación de Delaunay como representación geométrica. Sin embargo, nuestra propuesta no está limitada a construir representaciones triangulares como lo hace el método $Imesh$. En cambio, podemos modelar cualquier representación geométrica que pueda construirse a partir de una nube de puntos, como una malla de cuadriláteros o incluso, un diagrama de Voronoi, adaptado a la forma del objeto.

Para generar la triangulación de Delaunay, utilizamos la librería de geometría computacional *CGAL* (Fogel y Teillaud, 2015). En 2 dimensiones, esta librería tiene una complejidad algorítmica de $\mathcal{O}(\eta(log(\eta)))$, mientras que en 3 dimensiones su complejidad algorítmica es de $\mathcal{O}(\eta^2)$, donde η es el número de vértices de la triangulación. Como se ha podido mencionar, hasta ahora hemos usado métodos que no aumentan su complejidad algorítmica al ser extendidos de 2 a 3 dimensiones. Para acelerar la construcción de la triangulación de Delaunay, en 3 dimensiones, estamos usando la librería *GDel3D* de Cao et al. (2014), la cual argumenta ser alrededor de 10 veces más rápida que la librería *CGAL*.

En base a estas librerías, a continuación, se producen 2 mallas, las cuales están representadas por los símbolos $kMesh_{Cm}$ y $kMesh_{Im}$. La primera malla, es una representación geométrica sobre el borde que cumple con el segundo objetivo de nuestra propuesta, el cual se menciona en el Capítulo 1. La segunda malla, es una representación geométrica sobre el interior de la imagen, que se enfoca al tercer objetivo de nuestra propuesta, explicados a continuación.

4.3.1. Modelamiento con los puntos en el borde ($kMesh_{Cm}$)

A partir de los puntos distribuidos en el borde ($\mathcal{S}_{\mathcal{B}}$), en la etapa de Muestreo, construimos una primera representación geométrica simbolizada por $kMesh_{Cm}$, a través de la triangulación de Delaunay. Para construir esta representación, usamos los puntos distribuidos en el borde $\mathcal{S}_{\mathcal{B}}$ como generadores. De esta manera, construimos una representación geométrica que toma en cuenta los detalles del borde, considerando su curvatura y proximidad a otras partes del objeto. El resultado de este proceso, se puede observar en la Figura 4.12.

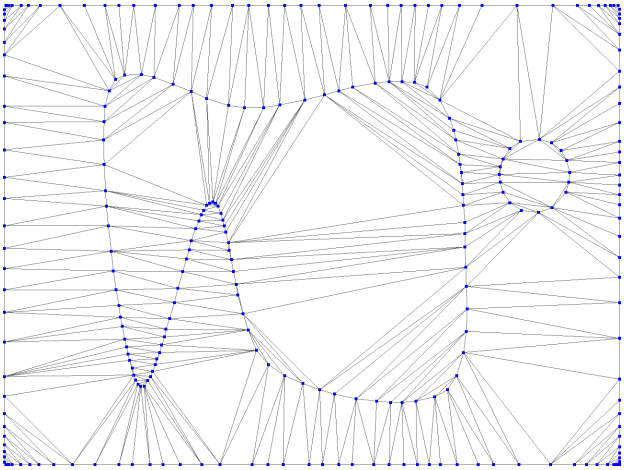


Figura 4.12: Representación geométrica usando los puntos distribuidos en el borde de la imagen en la etapa de Muestreo $\mathcal{S}_{\mathcal{B}}$, como vértices generadores en una triangulación de Delaunay.

Para tener una idea de la calidad de la triangulación generada, utilizamos un histograma de ángulos mínimos, como se puede ver en la Figura 4.13. De acuerdo a este histograma, y como es de esperarse, la malla $kMesh_{Cm}$ no presenta elementos con criterios de calidad, ya que la mayoría de elementos forma ángulos menores a 30 grados. Para mejorar la calidad de los elementos en la malla $kMesh_{Cm}$, proponemos construir una representación geométrica, utilizando el interior de la imagen, como se explica a continuación.

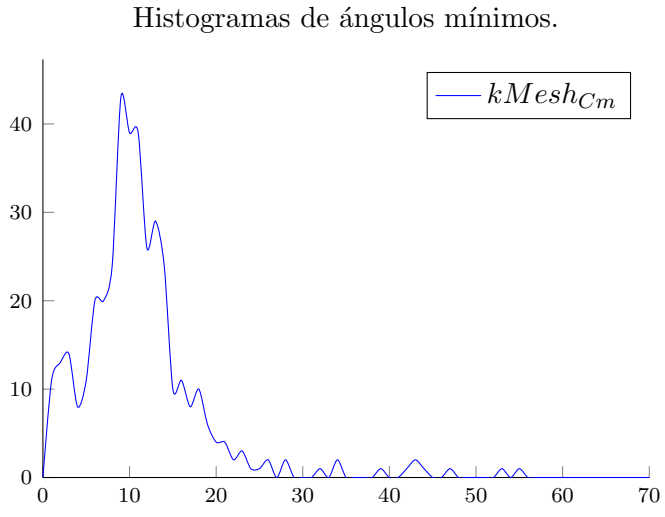


Figura 4.13: Histograma de ángulos mínimos generados por $kMesh_{Cm}$.

4.3.2. Modelamiento con los puntos en el interior ($kMesh_{Im}$)

Como se ha mencionado anteriormente, la malla $kMesh_{Cm}$, según el histograma de ángulos mínimos de la Figura 4.13, no presenta elementos de calidad. Para mejorar esto, proponemos una representación geométrica con los puntos distribuidos en el interior de la imagen ($\mathcal{S}_{\mathcal{I}}$). Como se ha visto, esta distribución de puntos considera el Mapa de densidad, por lo que se espera una distribución que genere una mejor representación geométrica, para producir una malla que contenga elementos con criterios de calidad, basados en la forma del objeto. El resultado de esta representación geométrica, se observa en la Figura 4.14.

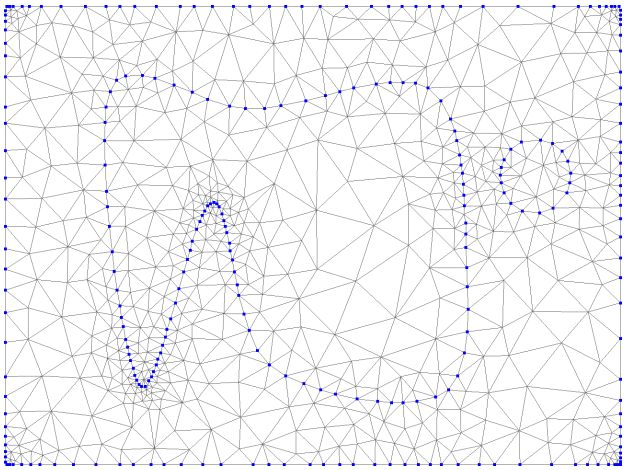


Figura 4.14: Representación geométrica usando los puntos distribuidos en el borde de la imagen en la etapa de Muestreo $\mathcal{S}_{\mathcal{I}}$, como vértices generadores en una triangulación de Delaunay.

De la misma manera que la malla $kMesh_{Cm}$, analizamos la malla $kMesh_{Im}$, para obtener el histograma de los ángulos mínimos formados por los elementos de la triangulación. La Figura 4.15, muestra el resultado de comparar el histograma generado por la malla $kMesh_{Cm}$ y la malla $kMesh_{Im}$. De acuerdo a este histograma, la malla $kMesh_{Im}$, presenta elementos con mejores criterios de calidad que la malla $kMesh_{Cm}$. Según este histograma, la mayoría de elementos que conforman $kMesh_{Im}$, están por encima de 20 grados.

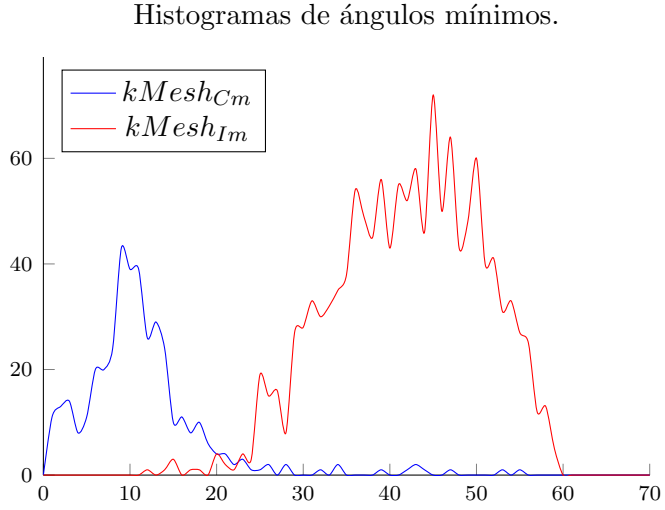


Figura 4.15: Comparación entre los histogramas de ángulos mínimos generados por los elementos de las mallas $kMesh_{Cm}$ y $kMesh_{Im}$.

En esta última etapa de nuestra propuesta, hemos utilizado los puntos distribuidos en la etapa de Muestreo para generar las representaciones geométricas $kMesh_{Cm}$y $kMesh_{Im}$. La calidad de los elementos generados por nuestro método, están basados en el Mapa de densidad de la primera etapa de nuestra propuesta. A diferencia del método $Imesh$, nuestra propuesta considera la forma del objeto como parte del proceso de construcción de la malla, para producir una representación geométrica que se adapta mejor a los detalles del borde.

En este Capítulo, se han descrito las 3 etapas de nuestra propuesta: Mapeo de densidad, la cual representa la imagen en base a un valor numérico que considera la curvatura y proximidad entre los elementos del borde; Muestreo, la cual distribuye puntos adaptados al Mapa de densidad; y Modelamiento, la cual genera una representación geométrica, utilizando los puntos de la etapa de Muestreo como vértices generadores, para producir las mallas $kMesh_{Cm}$ y $kMesh_{Im}$. Durante estas 3 etapas, se han presentado métodos que no aumentan su complejidad algorítmica cuando trabajan con imágenes en 3 dimensiones.

Como parte de la descripción realizada en este Capítulo, se ha hecho referencia a 3 parámetros requeridos por nuestra propuesta. El primer parámetro, es el conjunto de isovalores, utilizados para definir la frontera del objeto; el segundo parámetro, es el umbral de esqueletización, el cual es usado para extraer un esqueleto; y el tercer parámetro son los nuevos rangos de normalización del Mapa de densidad, para definir la distancia entre los puntos que van a ser distribuidos usando discos de Poisson. Estos parámetros van a ser discutidos a continuación, en el Capítulo 5, junto con algunos resultados obtenidos por nuestra propuesta a partir de imágenes en 2 y 3 dimensiones y los tiempos de ejecución de los métodos utilizados en nuestro trabajo.

Capítulo 5

Experimentos y resultados

En este Capítulo, se presenta y discute los resultados obtenidos aplicando nuestra propuesta, denominada $kMesh$, descrita en el Capítulo 4, para obtener las mallas $kMesh_{Cm}$ y $kMesh_{Im}$, a partir de imágenes en 2 y 3 dimensiones.

El resto de este Capítulo está estructurado de la siguiente forma. En la Sección 5.1, se detalla el *software* y *hardware* utilizado en el desarrollo de nuestra propuesta. En la Sección 5.2, se describen los intentos previos a la propuesta actual y sus influencias en esta. En la Sección 5.3, se presentan algunos resultados a partir de imágenes en 2 y 3 dimensiones, aplicando nuestra propuesta $kMesh$, para generar las mallas $kMesh_{Cm}$ y $kMesh_{Im}$. En la Sección 5.4, se discuten los parámetros de nuestro algoritmo. Finalmente, en la Sección 5.5, se muestran los tiempos de ejecución generados por los métodos utilizados en nuestra propuesta.

5.1. Descripción del *software* y *hardware* utilizado

Los experimentos presentados en este Capítulo han sido desarrollados en el lenguaje de programación C++ (Josuttis, 2012), en un procesador *4.2 GHz Intel Core i7* con 16GB de RAM y una tarjeta gráfica *GeForce GTX 1070*. Esta tarjeta gráfica tiene 15 multiprocesadores y 2048 *threads* por multiprocesador, es decir, un total de 30,720 *threads*. La programación paralela en *CPU* se ha implementado usando la librería OpenMP (Dagum y Menon, 1998) y la programación paralela en *GPU* se ha implementado con OpenCL (Stone et al., 2010). Las imágenes volumétricas fueron obtenidas de Roettger (2006).

En la implementación de nuestro algoritmo se ha utilizado la librería *ImageMagick* y *CImg* (Tschumperlé, 2012) para soportar distintos formatos de imágenes, como PNG, JPG, PGM. A su vez, en 2 dimensiones, utilizamos la librería de Geometría Computacional *CGAL* (Fabri y Teillaud, 2011), para generar la triangulación de Delaunay, en las 3 versiones de nuestro código. En 3 dimensiones, usamos la librería *gDel3d* (Cao et al., 2014), para producir la triangulación de Delaunay en la versión k^{gpu}. Mientras que en la

versión k_1^{cpu} y k_8^{cpu} en 3 dimensiones, utilizamos la librería *CGAL*.

A continuación, en la Sección 5.2, se describen los intentos que han sido realizados antes de la propuesta, descrita en el Capítulo 4.

5.2. Intentos anteriores a la propuesta actual

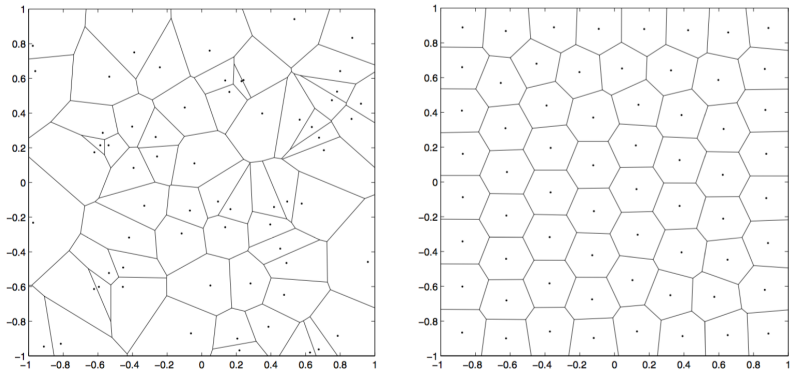
Para elaborar la propuesta presentada en el Capítulo 4, se realizaron diferentes intentos previos con el objetivo de distribuir puntos sobre una imagen y utilizarlos para construir una malla, a través de una representación geométrica, como la triangulación de *Delaunay*.

Debido a la dualidad que existe entre la triangulación de *Delaunay* y el diagrama de Voronoi el primer intento se dio por el lado de este último, como se describe más adelante en la Sección 5.2.1. Este primer intento, estuvo basado en el método de Lloyd para generar un diagrama de Voronoi centroidal (Du et al., 2006) y el método de aprendizaje GNG (Holmström, 2002), que fueron utilizados para distribuir puntos de forma uniforme sobre el borde ($\mathcal{B}$) y el interior ($\mathcal{I}$) de la imagen. Sin embargo, al distribuir puntos de forma uniforme, no se podía controlar la densidad de elementos sobre un área determinada de la imagen. Este motivo dirigió nuestra investigación a un segundo intento, descrito en la Sección 5.2.2, considerando la forma y distancia entre los elementos del borde, para distribuir un conjunto de puntos de forma adaptativa sobre la imagen. Este criterio de forma y distancia, en la propuesta actual, está expresado a través del Mapa de Densidad ($\mathcal{D}$), generado por la etapa de Mapeo de Densidad. Además, en este segundo intento, se utilizó sólo el método GNG para distribuir puntos sobre la imagen, debido a que el método de Lloyd requería generar un diagrama de Voronoi para un conjunto de puntos, en cada iteración del algoritmo. Es así que, finalmente, la propuesta actual, conservó el Mapa de Densidad ($\mathcal{D}$) y reemplazó el método GNG por el muestreo de discos de Poisson, en la etapa de Muestreo, para distribuir puntos sobre la imagen y utilizarlos como generadores en la triangulación de *Delaunay*.

5.2.1. Método basado en la distribución de puntos uniformes.

Al igual que la propuesta actual (Capítulo 4), se utilizó un conjunto de isovalores para definir el borde ($\mathcal{B}$) y el interior ($\mathcal{I}$) de una imagen. Sobre estas partes de la imagen, se empleó el algoritmo de Du et al. (2006) para generar un diagrama de Voronoi centroidal (Figura 5.1) sobre puntos seleccionados aleatoriamente.

Para obtener el diagrama de Voronoi centroidal a partir de un conjunto de puntos, en cada iteración se debe generar el diagrama de Voronoi usando los puntos como generadores, y por cada región de Voronoi obtenida, se actualiza el punto generador hacia el centroide de la región. Este proceso es repetido hasta que la diferencia entre el centroide y el punto generador alcance un error determinado, o también puede ser controlado por un número



(a) Diagrama de Voronoi con 64 puntos aleatorios.

(b) CVT. Diagrama de Voronoi centroidal generado con los puntos de (a).

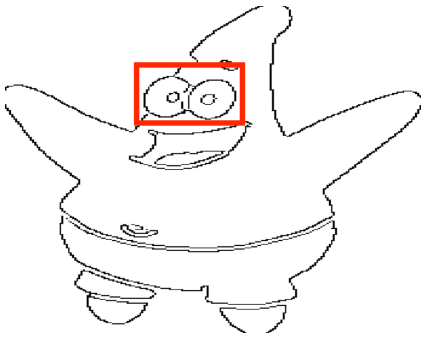
Figura 5.1: (Du et al., 1999). Generación del diagrama de Voronoi centroidal (CVT) en 2 dimensiones, a partir de 64 puntos aleatorios.

de iteraciones previamente definido.

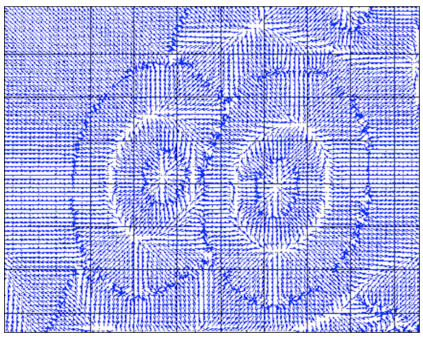
En nuestro primer intento, se seleccionó de forma aleatoria una cantidad de puntos, de acuerdo a un porcentaje del total de elementos, tanto para el borde como el interior de la imagen, como se observa en la Figura 5.2c. Sobre estos puntos aleatorios, se procedió a generar el diagrama de Voronoi centroidal, para el borde y el interior al mismo tiempo. Durante este proceso de relajación, algunos puntos generadores salían del borde y comenzaban a formar parte del interior. Para solucionar este problema, se utilizó el flujo de vector de gradiente (Xu y Prince, 2014) o GVF, el cual genera un mapa de vectores, en toda la imagen, que "apunta" hacia el borde, como se muestra en la Figura 5.2b. Así, si un punto salía del borde, regresaba a este, usando la información del GVF. La Figura 5.2d presenta el resultado de este proceso de relajación, basado en el diagrama de Voronoi centroidal, utilizando la corrección por el GVF.

En la Figura 5.2d se muestra la distribución final de puntos, sobre el borde y el interior de la imagen. Sin embargo, las zonas resaltadas, demuestran 2 problemas de este primer método. El primer problema, resaltado por el color negro, se debió a la densidad inicial de elementos. Por un lado, se podía controlar la cantidad total de puntos tanto sobre el borde como el interior, pero lo que no se podía hacer es definir la densidad de puntos para zonas específicas. Por esta razón, la zona resaltada, presenta menos cantidad de puntos. El segundo problema, resaltado por el color verde, se originó debido a la corrección por el flujo de la gradiente. Si un punto regresaba al borde usando el mapa de vectores, no consideraba a los puntos que ya pertenecían al borde, y podía, como se muestra en la parte resaltada, quedar un punto al lado del otro, o incluso sobre este. Como estos problemas se observaron en el borde, debido al espacio limitado que tenían los puntos para relajarse, se sustituyó el método basado en diagramas de Voronoi, por el método de aprendizaje GNG, para distribuir puntos sobre el borde de la imagen.

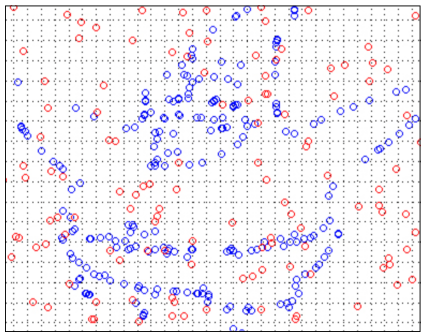
Debido a los problemas que han sido mencionados en el párrafo anterior, se utilizó el método GNG que nos permitió, al igual que el método de Lloyd, distribuir una cantidad definida de puntos. Durante el proceso del método GNG, un grafo es construido hasta contener la cantidad definida de vértices. En cada iteración, este método adapta de manera



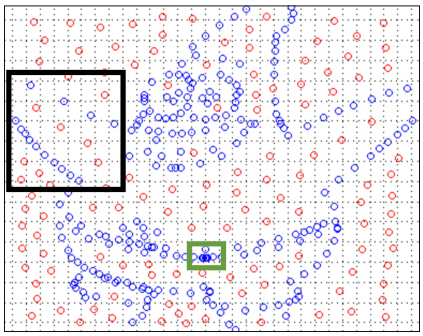
(a) *Patrick*. Borde (negro) e interior (blanco) de una imagen en 2 dimensiones de tamaño 904 x 710.



(b) Flujo de vector de gradiente para el área resaltada de (a).



(c) Puntos aleatorios seleccionados sobre el borde (azul) y el interior (rojo) de la imagen (a).



(d) Resultado de aplicar el método de Lloyd sobre los puntos del borde y el interior, usando la corrección por el flujo de vector de gradiente (b).

Figura 5.2: Método basado en el diagrama de Voronoi centroidal para distribuir puntos sobre el borde ($\mathcal{B}$) y el interior ($\mathcal{I}$) de la imagen, utilizando el flujo de vector de gradiente (GVF) para mover los puntos que salen del borde, de regreso hacia este. Las zonas resaltadas muestran algunos comportamientos no deseados de este primer intento.

uniforme, los vértices que conforman el grafo, hacia un conjunto de elementos, que en este trabajo, están conformados por el borde. Es así que, la Figura 5.3, muestra los resultados de utilizar este método para colocar puntos sobre el borde a una distancia uniforme, y el método de Lloyd para relajar los puntos aleatorios que se ubican en el interior de la imagen.

En esta Sección, se ha descrito el primer intento para distribuir puntos de forma uniforme sobre la imagen, resultando en el método GNG para colocar puntos en el borde y el método de Lloyd en el interior. A continuación, en la Sección 5.2.2, se describe el segundo método o intento propuesto, que toma en cuenta la forma del objeto para distribuir, de manera adaptativa, un conjunto de puntos sobre la imagen.

5.2.2. Método basado en la distribución de puntos adaptativos

Como se ha descrito en la Sección 5.2.1, los primeros intentos consistieron en distribuir un conjunto definido de puntos, separados por una misma distancia, sobre el borde ($\mathcal{B}$) y el interior ($\mathcal{S}_\mathcal{I}$) de la imagen. En este segundo intento, se incorporó la forma del